Les ManiaLinks de ManiaPlanet

Le didacticiel

Par *FT»Marcel*

Traduit de l'anglais par Tavernicole

Traduit de l'anglais le 20 avril 2017

Par Tavernicole

Préface

Les ManiaLinks sont de petits sites Web, ou des éléments graphiques et textuels qui peuvent être visualisés directement dans le jeu. Si vous avez toujours voulu créer les vôtres, vous êtes au bon endroit car ce didacticiel vous présente les ManiaLinks et leur création.

Vous savez déjà comment faire un ManiaLink pour TrackMania Forever, et souhaitez maintenant améliorer vos connaissances vers ManiaPlanet ? Là aussi, ce document vous y aide en vous donnant les informations vous permettant de mettre à niveau vos connaissances.

Ce didacticiel est constitué des cinq parties suivantes :

La première partie est destinée aux débutants qui n'ont jamais entendu parler de ManiaLinks et de leur création, et présente les connaissances de base nécessaires en XML pour construire un ManiaLink dans ManiaPlanet.

Si vous avez déjà appris comment faire un ManiaLink pour TrackMania Forever, commencez par la deuxième partie du didacticiel, qui décrit les différences entre ce que vous connaissez et ManiaPlanet.

La troisième partie contient une référence complète de tous les éléments utilisables dans un ManiaLink, et de tous les attributs de chaque élément, avec une description détaillée de ces derniers.

La quatrième partie vise plutôt les experts qui ont déjà construit des ManiaLinks et veulent désormais manipuler les éléments en utilisant le langage ManiaScript.

La cinquième et dernière partie de ce didacticiel décrit différents aspects des ManiaLinks, dont l'importance est loin d'être négligeable mais nécessite des connaissances avancées. Cette dernière partie peut être considérée comme une collection de sujets indépendants du reste de ce manuel.

Bien que les ManiaLinks dans ManiaPlanet soient pour la plupart identiques à ceux de TrackMania Forever, je ferai parfois la différerence entre les versions de ManiaLinks et les nommerai selon la version du jeu dans lesquels ils ont été introduits : "ManiaLinks de ManiaPlanet", "ManiaLinks de Forever" et leurs ancêtres "ManiaLinks d'United", les premiers arrivants, introduits dans TrackMania United, et ayant une structure complètement différente que nous décrirons dans ce document (bien que vous n'en aurez pas l'utilité) afin que ce didacticiel soit exhaustif.

Note du traducteur

Au-delà d'une simple transformation du texte en français, je me suis permis de modifier légèrement certaines parties du document, soit parce qu'elles étaient incomplètes ou absentes de la version originale, soit pour corriger des coquilles ou des incohérences.

J'espère que l'auteur original ne m'en voudra pas de m'être permis ces modifications, car elles n'ont été effectuées que pour apporter un petit plus à cet ouvrage déjà fort complet.

Sommaire

Introduction aux ManiaLinks de ManiaPlanet

Des bases aux détails des mécanismes internes

Le premier chapitre du didacticiel traite des bases des ManiaLinks. Si vous n'avez jamais écrit votre propre ManiaLink avant, c'est le bon endroit pour commencer. Ce chapitre commence par une courte description du XML, sur lequel les ManiaLinks sont basés. Les notions de base des ManiaLinks sont ensuite abordées, dont le système de positionnement permettant de correctement placer les éléments.

Si vous souhaitez rendre votre ManiaLink public, le dernier paragraphe de ce chapitre explique comment enregistrer un code de substitution pour l'URL complète de votre ManiaLink. Le seul prérequis à ce paragraphe est que vous sachiez et ayez la possibilité de télécharger des fichiers sur le Web dans un espace accessible au public.

Formation accélérée au XML et à sa syntaxe

Les ManiaLinks utilisent la norme XML pour spécifier les éléments à afficher. Si vous connaissez déjà le HTML, vous connaissez la syntaxe de base utilisée pour le XML et donc pour les ManiaLinks, mais il existe de petites différences entre le HTML et le XML, un peu plus strict.

Aussi, qu'est exactement le XML ? C'est l'acronyme de *eXtensible Markup Language* et est une syntaxe générale pour décrire divers documents. Bien que XML soit normalisé, il ne crée aucune restriction quand aux éléments à utiliser : l'application elle-même décide des éléments nécessaires et de ce qu'ils doivent représenter.

Dans cette partie du didacticiel, je ne décrirai que ces fonctionnalités de XML que vous devez vraiment savoir pour créer un fichier ManiaLink valide.

Hello World : Creation de votre premier ManiaLink.

Un document XML est simplement un fichier texte, dont le nom se termine généralement par .xml pour indiquer visuellement la nature de son contenu. Cela signifie que vous pouvez utiliser n'importe quel éditeur de texte pour le créer ou le modifier. Cependant, un éditeur avec fonctionnalité de coloration syntaxique est recommandé car cela permet de révéler directement la plupart des erreurs syntaxiques. On peut ainsi citer les logiciels suivants :

- Notepad++
- EmEditor
- Adobe Dreamweaver.

Après installation éventuelle et ouverture de notre éditeur, sélectionnons dans les menus ou les boîtes de dialogue de configuration notre désir de créer ou modifier un fichier XML afin d'activer la coloration syntaxique adaptée à ce type de fichier, puis entrons le code suivant :

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
    <label text="Hello World!" />
</manialink>
```

Au moment d'enregistrer notre fichier, nous devons nous assurer des deux points suivants :

1. Le fichier doit avoir le bon encodage. Ceci est important, notamment si nous utilisons des caractères spéciaux comme les lettres accentuées, le "ß" allemand, ou les caractères est-asiatiques. Le jeu de caractères réglé par défaut dans notre éditeur, "europe de l'ouest", "Latin-1" ou "ISO-8859-1" provoquera l'affichage de caractères bizarres dans ManiaPlanet. Nous devons enregistrer notre fichier en "UTF-8 sans signature".

2. Le fichier doit avoir la bonne extension : ".xml". Ceci permet, lors de l'ouverture ultérieure du fichier dans notre éditeur, d'activer automatiquement la coloration syntaxique adéquate.

Sauvegardons notre fichier sous le nom "hello.xml" dans le dossier "Media\ManiaLinks" situé dans notre espace personnel de stockage de ManiaPlanet (par défaut "Mes documents\ManiaPlanet")

On lance ManiaPlanet et entrons : *files://Media/ManiaLinks/hello.xml* dans la barre d'adresse qui s'affiche lorsqu'on place la souris en haut de l'écran. Si on voit "Hello World" écrit en blanc sur fond gris, on a créé notre premier ManiaLink !

Fermons le navigateur intégré à ManiaPlanet en cliquant sur le bouton situé à gauche de la barre d'adresse ou en appuyant sur la touche Échap.

Note

> Le protocole d'accès *file://* ne peut fonctionner que localement sur notre machine, personne d'autre que nous ne peut visualiser notre ManiaLink. Nous devrons téléverser ce fichier sur l'internet pour pouvoir l'appeler au moyen du protocole d'accès *http://* comme pour tout site web.

Mais avant de voir comment créer un code pour un ManiaLink, nous devons d'abord parler de grammaire XML.

Un examen plus approfondi de la syntaxe : l'en-tête XML

Si vous observez l'exemple ci-dessus, vous voyez en première ligne l'en-tête XML. Ce dernier introduit toujours un fichier XML et indique à l'application que le reste du fichier est de l'XML. Et les valeurs spécifiées dans cet en-tête définissent en quelque sorte la configuration du contenu du fichier.

Cette ligne est pratiquement toujours la même, donc oubliez les valeurs de configuration et copiez/collez cette ligne dans chaque nouveau fichier XML que vous créez, puisqu'actuellement il n'existe aucune autre version que la 1.0, et l'encodage UTF-8 est le meilleur choix pour que votre ManiaLink soit compatible avec toutes les langues.

Note

> Il est fondamental que le premier caractère de votre fichier soit un "<". Vous ne pouvez en aucun cas mettre ni espaces, ni fins de lignes, ni d'autres caractères avant ce "<".

Tout ce qui est ouvert doit être refermé, surtout les tags

Qu'en est-il des trois autres lignes de l'exemple ? Il s'agit de code XML, décrivant le ManiaLink que nous voulons afficher dans ManiaPlanet.

Elles se composent de balises (*tags* en anglais) qui commencent toujours par un symbole "inférieur" et se terminent par un symbole "supérieur". Le premier terme situé après le caractère "<" nomme la balise, ce qui permet d'indiquer ce à quoi elle sert. Les autres termes appartiennent à la balise et donnent des indications complémentaires. Dans notre exemple, nous avons deux balises *manialink* et une balise *label*.

De plus, les balises peuvent inclure d'autres balises. Il est donc essentiel de refermer une balise après usage. Ceci se fait en précisant une balise de fin qui s'écrit comme celle de début, mais avec un caractère "/" précédant le nom de balise. On peut aussi refermer la balise dans la balise de début. Cela nous donne quatre types de balises :

- **La balise de début**, qui ne commence ni ne se termine par un "/", et n'est jamais seule. Elle doit être suivie, un peu plus loin, par une balise de fin. C'est le cas dans notre exemple de notre balise *manialink* située en deuxième ligne.
- **La balise de fin**, qui porte le même nom que la balise de début, précédé par un "/". Elle indique où se termine la balise et donc son contenu. Dans notre exemple, c'est le cas de la balise de fin *</manialink>* située en quatrième ligne. Toute balise de fin doit automatiquement se référer à la balise de début de même nom, la plus proche d'elle dans le code qui précède.
- **La balise autonome**, qui s'ouvre et se referme en une seule opération. Elle commence comme une balise de début mais comporte un caractère "/" juste avant le symbole ">". Cette écriture "/>" indique que la balise se termine à cet endroit. Bien entendu, une telle balise ne peut en contenir d'autres. C'est le cas de la balise *label* située en troisième ligne. Elle est équivalent à l'écriture : *<label text="Hello World!"></label>*
- **La balise de commentaire**, que nous verrons un peu plus loin. Elle ne comporte pas de caractère "/" ni en début, ni en fin.

De plus, une balise d'ouverture ou une balise autonome peut contenir un ou plusieurs *attributs*, chacun d'eux étant spécifié sous la forme

nom="valeur" (valeur entre guillemets) ou *nom='valeur'* (valeur entre apostrophes). Les balises de fin ne portent jamais d'attributs.

Ainsi, dans notre exemple, la balise d'ouverture *manialink* contient-elle un attribut nommé *version* ayant comme valeur *1*.

De même, la balise autonome *label* contient aussi un attribut, nommé *text* et dont la valeur est *Hello World!* que nous avons vu s'afficher.

Une balise de début et la balise de fin associée s'appellent un nœud du document XML (*node* en anglais). Tout ce qui se trouve entre les deux est le contenu du nœud. Ce contenu peut être du texte ou d'autres nœuds. Cependant, il n'est pas possible que les nœuds se chevauchent. Tout nœud commencé doit se terminer *avant* de terminer le nœud dont il fait partie. En d'autres termes, aucune balise de fin ne correspondant pas à la balise de début qui la précède n'est tolérée, et toute balise de fin se rapporte toujours à la balise de début qui la précède. La balise englobante s'appelle "nœud parent", la balise englobée s'appelle "nœud enfant".

Autorisé	**Interdit**
<parent> <enfant> <petitenfant> </petitenfant> </enfant> <enfant> </enfant> </parent>	<parent> <enfant> </parent> </enfant>

Dans notre exemple, *label* est le nœud enfant du nœud *manialink*. Le nœud *label* étant autonome, il ne peut avoir aucun nœud enfant.

Balises particulières : le nœud racine, les commentaires et l'entête

Le nœud *manialink* est un nœud spécial : il n'a aucun parent et, par conséquent, s'appelle le *nœud racine*. Un document XML valide doit toujours avoir un et un seul nœud racine, et qu'il contient par conséquent toutes les données du document en tant que nœuds enfants, on l'appelle aussi le *nœud du document*. Dans ManiaLinks, le nœud racine est toujours un <manialink>.

Un autre type spécial de balise est le commentaire. Vous pouvez placer des commentaires en lieu et place de tout autre nœud, donc évitez de le placer à l'extérieur du nœud racine, ni à l'intérieur d'une balise, entre les attributs. Comme son rôle l'indique, vous pouvez ajouter des commentaires supplémentaires à votre fichier XML, par exemple Pour indiquer la finalité d'une partie du fichier. L'analyseur XML, par définition, ignore tout commentaire dans le fichier. La syntaxe d'un commentaire est légèrement différente de celle des autres tags:

Code

```
<!-- Voici un commentaire -->
```

Un commentaire commence toujours par le symbole "<" suivi par un point d'exclamation, exactement deux tirets et un espace. Il se termine par un espace, deux tirets et le symbole ">". Entre le début et la fin de balise, vous pouvez utiliser tous les caractères que vous souhaitez, à l'exception notable de "--" (deux traits d'union) qui pourraient être interprétés comme la fin du commentaire, avec une erreur de syntaxe puisque non suivis du caractère ">".

La dernière balise particulière est l'entête. Elle est toujours située avant le nœud racine, en début de fichier. C'est la seule balise à pouvoir se trouver en dehors du nœud racine.

Les caractères spéciaux et leur échappement

Quelque soit le langage, à partir du moment où des caractères définissent une syntaxe, on a toujours le problème : imprimer ces caractères eux-mêmes. En XML, les caractères "inférieur" et "supérieur" et les guillemets perturbent la syntaxe XML, ce qui rend la totalité du fichier invalide. Bien sûr, il existe une façon de restituer ces entités en toute sécurité, c'est ce qu'on appelle l'échappement. Cinq caractères sont affectés par ce problème et s'échappent ainsi :

- `<` peut être remplacé par **`<`** (Lower Than)
- `>` peut être remplacé par **`>`** (Greater Than)
- `"` peut être remplacé par **`&quot;`** (Quote)
- `'` peut être remplacé par **`&apos;`** (Apostrophe)
- `&` peut être remplacé par **`&`** (Ampersand)

La séquence d'échappement commence toujours par un "et commercial" (ce qui rend ce caractère vulnérable), suivi d'un terme et d'un point-virgule. Avec cette écriture, vous avez la possibilité de placer, par exemple, un caractère ">" à l'intérieur de l'attribut d'une balise sans que l'analyseur XML ne détecte la fin de la dite balise. Une exception cependant : inutile d'échapper le texte des commentaires, puisqu'ils sont ignorés par l'analyseur.

Note

Si vous générez un ManiaLink au moyen de PHP et souhaitez afficher la valeur d'une saisie utilisateur, assurez-vous d'échapper ces caractères ! Sinon, un simple guillemet ou une apostrophe pourra rendre votre ManiaLink invalide.

Par exemple, si vous utilisez les apostrophes pour les valeurs d'attributs, vous pouvez utiliser la fonction htmlspecialchars() de PHP comme ceci :

```
$escapedText = htmlspecialchars($textToBeEscaped,
                                ENT_QUOTES,
                                'UTF-8');
```

Un petit test : trouvez les erreurs

Vous pensez être en mesure d'écrire un fichier XML valide ? Examinez le code suivant. Pouvez-vous débusquer toutes les erreurs qui s'y cachent ?

Code

```
<root>
   <!-- This is invalid XML -- Do not copy&paste! -->
   <child1><child2>Some content</child1></child2>
   <input type="checkbox" checked>
   <label link="http://www.example.org/?p1=v1&p2=v2" />
</root>
<standalone />
```

1. L'entête XML est absent

Bien que l'absence de l'en-tête XML n'empêche pas ManiaPlanet ou d'autres analyseurs XML de traiter correctement le fichier, ce n'est pas la façon correcte de définir un fichier XML. De la même manière que si vous

nommez une image jpeg avec .img, certains programmes n'ont aucun problème pour afficher l'image là où d'autres n'y arrivent pas, comme Windows lui-même qui est alors incapable d'en afficher la miniature.

2. Le nœud racine n'est pas seul

Le premier nœud d'un document (sauf l'en-tête XML) est toujours le nœud racine. Dans notre exemple, il s'agit de <*root*>. Un fichier XML valide ne peut contenir qu'un seul nœud racine, et la balise <*standalone*> n'est donc pas autorisée à figurer en dehors de <*root*>.

3. Entrelacement incorrect des nœuds *child1* et *child2*

Les nœuds <*child1*> et <*child2*> ne sont pas correctement imbriqués : aucune de ces balises ne contient complètement l'autre. Vous devez soit fermer <*child2*> avant de fermer <*child1*>, soit ouvrir <*child2*> avant d'ouvrir <*child1*> pour avoir un nœud englobant et un nœud englobé. Vous pouvez encore fermer <*child1*> avant d'ouvrir <*child2*> pour les rendre les deux balises enfants de la balise racine.

4. Absence de fermeture du nœud input et attribut sans valeur

La balise <*input*> n'est pas une balise autonome et il manque la balise de fin. De plus, l'attribut *checked* n'a pas de valeur. Cette balise montre les différences entre XML et HTML : alors que cette valise pourrait être valide en HTML, ce n'est pas le cas du XML.

5. Caractère "et commercial" non échappé

Le caractère `&` doit toujours être remplacé par `&`, même s'il s'agit d'un URL figurant dans la valeur d'un attribut d'une balise (ici l'attribut *link* de la balise <*label*>). Aucune exception. Sauf dans les commentaires.

6. Deux tirets à la suite dans un commentaire

Vous n'avez pas la possibilité d'écrire deux tirets de suite dans un commentaire, car ces derniers indiquent le début de la terminaison du commentaire. Par contre, rien ne vous oblige à échapper le `&`…

Une petite astuce pour terminer : si vous voulez vérifier la validité d'un fichier XML, vous pouvez l'ouvrir avec Firefox ou Chrome qui vont reconnaître un document XML et tenter d'afficher l'arbre analysé. Et qui vous préviendront en cas d'erreur.

Les bases du positionnement des éléments d'un ManiaLink

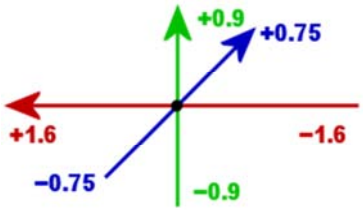
Avant que nous ne puissions réellement créer des ManiaLinks fonctionnels, nous avons besoin d'en savoir plus sur le système de positionnement des éléments.

Si tous les éléments visibles peuvent être placés comme on le souhaite sur l'écran sans avoir à se soucier des autres éléments, sans une bonne connaissance du système de mesure, il est difficile d'obtenir un résultat cohérent.

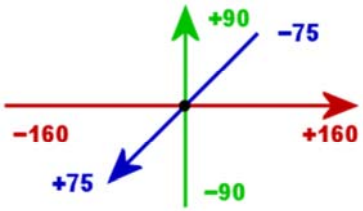
Les deux systèmes de positionnement

En gros, vous disposez d'un système de coordonnées tridimensionnelles complet, avec un axe X de gauche à droite, un axe Y de bas en haut, et un axe Z pour la profondeur et le placement des éléments sur ou sous les autres.

Pour rendre les choses un peu plus compliquées, vous avez le choix entre deux systèmes de coordonnées différents : ils sont tous deux aussi puissants l'un que l'autre et vous n'avez aucun effet indésirable que vous utilisiez l'un plutôt que l'autre. Les seules différences concernent la granularité des axes et leur orientation :



Positionnement classique (p-system)



Positionnement de menus (n-system)

La conversion entre ces deux systèmes est très simple, Il suffit de diviser ou de multiplier les valeurs par 100, en accordant une attention particulière aux axes X et Z inversés.

Il est cependant recommandé d'opter pour l'un de ces deux systèmes et de s'y tenir dans l'ensemble du ManiaLink.

La différence dans les ManiaLinks (sauf pour ce qui concerne les valeurs en elles-mêmes) est très faible : si vous souhaitez utiliser le

positionnement classique, vous devez utiliser les attributs *pos* et *size*, alors que pour utiliser le positionnement de menus, vous devez utiliser les attributs *posn* et *sizen*. Aussi, à chaque fois qu'un élément n'est pas placé là où vous l'attendez, vérifiez que vous utilisez les bons attributs et la bonne orientation. En outre, vérifiez que *version="1"* est spécifié dans la balise *<manialink>* d'ouverture, cela influençant également les valeurs des axes.

La valeur Z spécifie quel élément est couvert par les autres. Plus la valeur est élevée (système de menus) ou basse (système classique), plus l'élément est en avant-plan. Faites néanmoins attention à ces valeurs sur l'axe Z : si deux éléments ont les mêmes valeurs, ou si un élément a une valeur invalide (trop grande ou trop faible), l'ordre des éléments sera chamboulé, deviendra aléatoire, et pourra même changer par exemple en déplaçant la souris dans la barre supérieure de ManiaPlanet. Donc, assurez-vous toujours d'avoir des valeurs Z différentes si des éléments se chevauchent.

Pourquoi avoir deux systèmes de coordonnées différents, alors qu'un seul suffirait ?

La raison en est principalement historique : le système de positionnement classique a été utilisé dans la première version des ManiaLinks (ManiaLinks d'United), puis a été revu pour les ManiaLinks de Forever en tant que système distinct, principalement car le système classique n'était pas très adapté aux menus du jeu TrackMania Forever. Cependant, par mesure de portabilité ascendante, l'ancien système a été conservé afin que des ManiaLinks d'United fonctionnent toujours dans ManiaPlanet. Et pour faciliter les choses, les valeurs min/max des deux systèmes ont été rendues plus similaires lorsque l'écran large 16: 9 a été introduit dans ManiaLinks, de sorte que la conversion entre les systèmes n'est désormais plus un problème.

Faisons une pause : premiers exemples de positionnement

Nous en avons presque terminé avec les bases, mais avant d'aborder la dernière notion, l'alignement des éléments, nous allons voir quelques exemples de positionnement.

Dans ces exemples, nous allons utiliser des Quads pour afficher les positions, car nous avons la possibilité de colorer l'arrière-plan des Quads afin de les rendre visibles (nous reviendrons plus tard sur le détail des éléments à notre disposition pour créer des ManiaLinks).

Commençons par un exemple très basique :

Code

```
<quad bgcolor="F00A" />
```

Nous n'avons ni spécifié de position, ni de taille pour ce Quad. Seule la couleur d'arrière-plan a été fournie pour le rendre visible. Néanmoins, cela va afficher un rectangle coloré, car ManiaPlanet applique des valeurs par défaut à tous les attributs manquants. Les deux lignes suivantes produiront exactement le même Quad que celui ci-dessus, à la différence que nous avons explicitement indiqué position et taille :

Code

```
<quad posn="0 0 0" sizen="100 100" bgcolor="F00A" />
<quad pos="0 0 0" size="1 1" bgcolor="F00A" />
```

Deux choses sont mises en valeur par cet exemple : d'abord, les deux systèmes de positionnement se valent. Tout ce que nous spécifions dans le système de menus (premier Quad), nous pouvons le convertir au système classique (deuxième Quad). C'est pourquoi désormais les exemples n'utiliseront plus que le système de menus (quoique vous puissiez utiliser le système classique si vous préférez). Ensuite, cela met en avant les valeurs par défaut pour les attributs de position, à savoir 0 sur chaque axe, et les valeurs par défaut pour le dimensionnement, à savoir 1.

Code

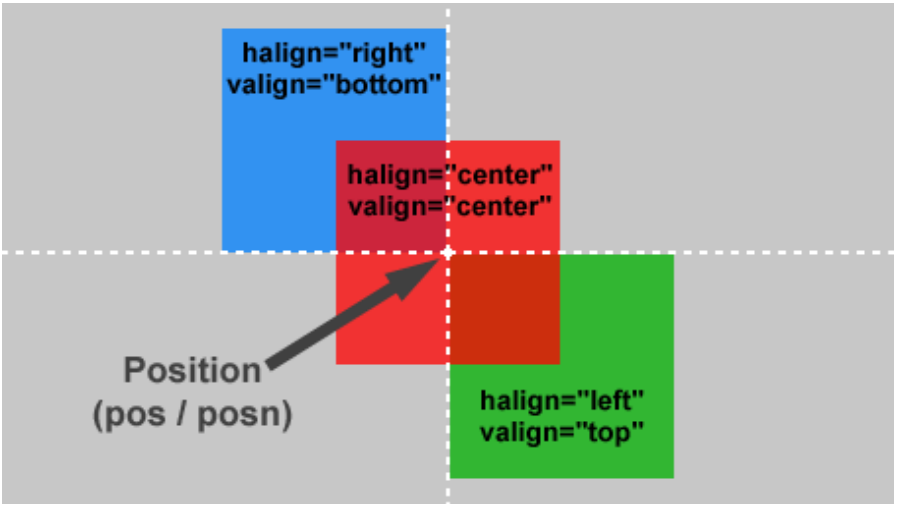
```
<quad posn="40 -40" sizen="20 20" bgcolor="F00A" />
```

Si nous n'avons pas besoin de définir une la valeur Z spécifique, elle peut être omise et ne spécifier que les valeurs pour X et Y. Et avec des valeurs différentes de zéro, ce Quad ne sera plus au milieu de l'écran. En se reportant au système de coordonnées de menus, on peut constater que le rectangle est désormais décalé vers le bas et à droite de l'écran.

Pour un peu plus de confusion : l'influence de halign et valign

Une autre notion à inclure dans le contexte du positionnement des éléments est leur alignement horizontal (*halign*) et vertical (*valign*). Le

schéma suivant montre trois Quads pour lesquels les valeurs des attributs posn sont identiques, mais pour lesquels les valeurs des attributs *halign* et *valign* ont été définis de manières différentes :



L'alignement semble être source de confusion. L'élément aligné à gauche est à droite de celui aligné à droite… Et le Quad aligné en bas est au-dessus de celui aligné en haut…

En fait, les alignements ne correspondent pas à la position sur l'écran, mais au positionnement du Quad par rapport au point de référence représenté par la coordonnée spécifiée par *pos* ou *posn*.

Un Quad aligné à droite indique que ce dernier considère la position comme sa marge droite. Un Quad aligné à gauche considère la position comme sa marge gauche. Et il en est de même avec les alignements verticaux.

Par défaut, la valeur de *halign* est *left* et la valeur de *valign* est *top*, et en conséquence, les Quads sont placés à droite et en-dessous du point de référence représenté par les valeurs X et Y de *pos* ou *posn*.

Enregistrement d'un code pour un ManiaLink

Il est possible de créer un code pour son ManiaLink. Cela revient en quelque sorte à créer un raccourci. Avec ce dernier, le joueur pourra accéder à votre page sans devoir saisir dans la barre d'adresse de

ManiaPlanet un long URL avec protocole, nom de domaine, chemin et nom de fichier XML. Il lui suffira de saisir le raccourci pour se retrouver au sein de votre œuvre.

Vous pouvez essayer cela très simplement : Dans ManiaPlanet, placez votre souris en haut de l'écran pour faire apparaître la barre d'outils, dont la partie gauche est dédiée à la saisie d'un URL ou d'un code de ManiaLink. Cliquez dans cette partie gauche, entrez *styles* et validez avec la touche ENTREE. Vous voilà dans un ManiaLink dédié aux concepteurs de ManiaLinks et listant les styles et sous-styles disponibles.

En réalité, le code *styles* est un raccourci vers un serveur web publiant des ManiaLinks, c'est-à-dire des fichiers XML sujets du présent didacticiel.

Créer un code pour un ManiaLink

Cette opération est très simple à effectuer : connectez-vous à votre page joueur depuis votre navigateur internet préféré (IE, Firefox, Chrome, Safari, etc) en y saisissant l'URL : **https://player.maniaplanet.com** et identifiez-vous avec votre login et votre mot de passe.

Dans la barre de menus supérieure, sélectionnez *Advanced* et cliquez sur la commande de menu *ManiaLinks*.

La page qui apparaît alors donne la liste des codes que vous avez créé, surmontés d'un formulaire de création de code de ManiaLink.

Renseignez la zone de saisie "Code" avec le raccourci que vous souhaitez réserver. Entrez un terme représentant le contenu de votre ManiaLink. Notez cependant que votre choix est restreint par le fait que :

- Votre code doit être unique : il ne peut exister deux codes identiques dans l'univers ManiaPlanet, et la règle habituelle sur le net du premier arrivé, premier servi, s'applique.
- Votre code ne doit pas être contestable par autrui, Nadeo se réservant le droit de modifier ou supprimer tout code objet d'une réclamation.

Dans la zone de saisie *URL of the XML file*, entrez l'URL complet de votre ManiaLink. Cet URL doit être accessible au public. Vous devrez donc éviter les URL de type "http://localhost", "http://127.0.0.1" ainsi que les adresses IP de réseaux privés spécifiées par la RFC 1918 (ex: "http://192.168.x.y").

Entrez votre code de validation dans la zone *Validation code* afin de confirmer votre identité et cliquez sur le bouton *Save this code*.

Après vérification des informations saisies et si tout est ok, votre code et l'URL associé apparaît dans le tableau inférieur : c'est fait, votre code de ManiaLink est créé et directement utilisable, soit dans les attributs *manialink* des balises (que nous verrons un peu plus loin), soit en tant qu'adresse du point d'entrée de vos ManiaLinks, que vous pourrez diffuser auprès de vos amis et connaissances, exactement comme le ManiaLink *Styles* que nous avons vu précédemment.

Différences avec les ManiaLinks de Forever
Tout ce qui a été modifié dans ManiaPlanet

Ce chapitre du didacticiel est pour tous ceux qui connaissent déjà les ManiaLinks de Forever et souhaitent "mettre à niveau" leurs connaissances vers les ManiaLinks de ManiaPlanet. Si vous venez de travailler avec les bases de ManiaLinks, passez au chapitre suivant, car celui-ci ne contiendra aucune information nouvelle dont vous pourriez avoir besoin.

Fondamentalement, aucun élément majeur n'a changé de Forever vers ManiaPlanet : si vous avez un ManiaLink Forever, il fonctionnera dans ManiaPlanet sans aucune modification. Néanmoins, il y a quelques détails que vous devriez connaître concernant les ManiaLinks de ManiaPlanet…

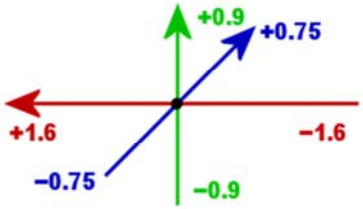
Les ManiaLinks de version 1

La principale différence avec les ManiaLinks de Forever est la nouvelle version qui a été introduite avec ManiaPlanet, qui permettra ou modifiera le comportement de certaines fonctionnalités pour les préparer à l'avenir. Alors que les ManiaLinks de Forever (et leurs ancêtres les ManiaLinks d'United) sont désormais appelés à se voir attribuer la version 0, les ManiaLinks de ManiaPlanet deviennent la version 1. Ce numéro de version est simplement spécifié dans la balise de début *<manialink>*, et de ce nouveau numéro de version découlent deux caractéristiques majeures :

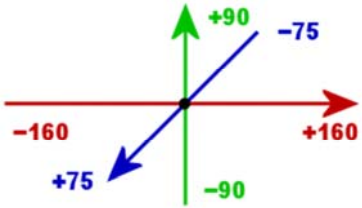
Le support du 16:9

La version 0 des ManiaLinks dispose d'un système de coordonnées basé sur un écran 4:3. Le développement des moniteurs ces dernières années impose désormais la prise en compte d'écrans aux ratios plus étendus dont le 16:9 qui s'est généralisé.

En définissant la version sur 1, vous activez un nouveau système de coordonnées optimisé pour les écrans panoramiques 16:9, et les nouvelles valeurs des axes des systèmes de positionnement sont désormais les suivants :



Positionnement classique



Positionnement de menus

La différence principale est la modification du rapport entre les deux systèmes de mesure, plus simple, puisque d'un rapport de 64 entre le positionnement classique et le positionnement de menus, c'est désormais un facteur de 100 qui les distingue, d'où une simplification de la conversion d'un positionnement à l'autre.

Bien que ManiaScript fonctionne avec la version 1 et la version 0, il utilise toujours le système de coordonnées 16:9 pour calculer les positions. Par conséquent, assurez-vous d'utiliser toujours la version 1 lorsque vous utilisez un ManiaScript afin d'éviter toute confusion

Plus de support de la syntaxe ligne/cellules

Le passage en version 1 des ManiaLinks désactive toute prise en charge des ManiaLinks d'United, également appelée syntaxe de ligne/cellule. Ce changement n'est pas très dramatique, car les ManiaLinks d'United sont obsolètes depuis la mise à jour en TrackMania Forever et n'est plus utilisé depuis plusieurs années. Si vous ne savez pas ce qu'est la syntaxe ligne/cellule, ne vous en inquiétez pas, vous n'en n'aurez plus besoin.

Si vous développez un nouveau ManiaLink pour ManiaPlanet, il est recommandé d'utiliser la version 1 pour rendre votre ManiaLink prêt pour l'avenir. Avec la nouvelle version, les coordonnées sont simplifiées et correspondent mieux aux écrans larges devenus désormais la norme.

Les nouveaux attributs et ceux qui ont changé

Que vous utilisiez la version 0 ou 1, presque tous les éléments ont maintenant de nouveaux attributs. Si à ce stade du didacticiel, une

courte liste de tous les nouveaux attributs devrait être donnée, pour plus de détails, consultez la référence dans la partie suivante.

Liste des attributs ajoutés et modifiés

- L'attribut *version* de la balise <*manialink*> spécifie désormais la version du ManiaLink.
- Le nouvel attribut *background* de la balise <*manialink*> permet d'avoir un arrière-plan transparent.
- La nouvelle valeur *center2* pour l'attribut *valign* des éléments de text aligne le texte légèrement différemment par rapport à la valeur *center*.
- Les valeurs des attributs *style* et *substyle* de toutes les balises autorisant les styles ont évolué. Référez-vous au Manialink "styles" pour obtenir la liste complète.
- Le nouvel attribut *id* a été ajouté à tous les éléments positionnables autorise l'accès aux éléments depuis les ManiaScripts.
- Le nouvel attribut *ScriptEvents* ajoutés aux éléments visibles active le support des évènements via les ManiaScripts.

L'intégration du ManiaScript

L'une des grandes nouveautés de ManiaPlanet, et le plus grand changement par rapport aux ManiaLinks, est l'ajout des ManiaScripts. Alors que les ManiaLinks de Forever étaient totalement statiques et immuables après leur affichage dans le jeu, les ManiaScripts permettent désormais d'en accroître la dynamique. Vous êtes maintenant en mesure de réagir sur les actions effectuées par les utilisateurs sans devoir recharger le ManiaLink : des événements sont déclenchés lors de la manipulation de la souris ou du clavier.

Ainsi, si l'utilisateur déplace le pointeur de la souris sur une icône, une bulle d'aide peut s'afficher. Vous pouvez de même placer un bouton permettant de fermer une boîte de dialogue… Sans actualiser le ManiaLink. Et ce ne sont que des exemples très basiques de ce qu'il est enfin possible de faire.

Vous en voulez plus ? Lisez le chapitre concernant les ManiaLinks…

Le remplacement de AddPlayerID par ManiaConnect

Dans TrackMania Forever, vous aviez la possibilité d'utiliser l'attribut *addplayerid* dans certains éléments pour récupérer des informations sur l'utilisateur du ManiaLink. Cette fonctionnalité n'est plus disponible dans ManiaPlanet, principalement parce qu'il était trop facile de manipuler les valeurs soumises, ce qui selon Nadeo, n'est pas compatible avec le concept ManiaPlanet.

En remplacement de l'attribut *addplayerid*, Nadeo a introduit un nouveau mécanisme appelé ManiaConnect pour accéder aux données du joueur.

Avec ManiaConnect, le joueur doit explicitement confirmer qu'il donne son accord pour publier les données sur un ManiaLink donné, et ce n'est que lorsque cet accord a été donné que le ManiaLink peut récupérer les données du joueur. Ce nouveau mécanisme est complètement sécurisé et ne peut pas être manipulé, ce qui ouvre de nouvelles possibilités : désormais rien n'empêche par exemple un panneau administrateur d'accorder des droits spécifiques à un joueur via les informations fournies par ManiaConnect. C'est ce dernier qui authentifie le joueur et fournit l'information au ManiaLink.

Néanmoins, cette sécurisation se paye en rendant cet accès à l'information un peu plus complexe en utilisant les WebServices de ManiaPlanet.

Vous trouverez plus avant un exemple complet d'authentification avec ManiaConnect.

Autres modifications

L'accès aux fichiers locaux

Une autre nouvelle fonctionnalité est la possibilité d'accéder directement aux fichiers stockés sur l'ordinateur local à l'aide du protocole `file://`. Le répertoire de base est le dossier personnel de ManiaPlanet, par défaut situé dans "Mes documents".

Ainsi, le fichier Mes Documents\ManiaPlanet\Media\ManiaLinks\test.xml est-il accessible avec la notation "file://Media/ManiaLinks/test.xml". à titre d'exemple, essayez "file://Media/ManiaLinks/Solitaire/index.xml" fourni dans l'un des packs livrés avec le jeu...

Cette fonctionnalité est particulièrement utile lors du développement d'un ManiaLink : au lieu d'avoir à le téléverser à chaque fois sur le serveur, vous pouvez l'appeler directement à partir de votre disque dur, à condition que le fichier soit placé dans le dossier Media \ ManiaLinks. Bien sûr, lors de la publication d'un ManiaLink, toutes les références locales doivent être supprimées et remplacées par des références en ligne.

Le support des formats vidéo

ManiaPlanet ne supporte plus officiellement le format Bink (fichiers .bik). Cependant, ManiaPlanet supporte à la place le format WebM.

Modification du UserAgent et des entêtes HTTP

ManiaPlanet n'utilise plus la valeur *Gamebox* pour le UserAgent, remplacé par une valeur plus détaillée comme par exemple "ManiaPlanet/3.0.0 (2011-09-14)". Ainsi, en plus de permettre l'identification du jeu lors de l'appel d'un ManiaLink, les informations de version sont enfin disponibles. Cela s'avèrera utile au fur et à mesure de l'enrichissement des ManiaLinks afin d'adapter la réponse du serveur aux capacités du logiciel de l'utilisateur.

De plus, l'entête http contient désormais l'information *Accept-Language*, spécifiant la langue de l'utilisateur, ce qui permet une traduction automatique de vos ManiaLinks pour peu que vous ayez renseigné la version locale de vos ressources. Reportez-vous à la balise <*dico*> pour plus d'informations.

Les éléments des ManiaLinks
La référence complète de tous les éléments

Ce chapitre contient la référence complète de tous les éléments des ManiaLinks de ManiaPlanet, incluant la liste détaillée des attributs pris en charge. Si vous débutez, vous voudrez peut-être lire une fois cette partie du didacticiel pour avoir un petit aperçu des éléments existants et de leur utilité. Le but recherché, néanmoins, est d'établir une référence : inutile de vous souvenir de l'ensemble des détails, vous pourrez y revenir par la suite pour y trouver l'information ponctuelle qui vous fait défaut.

ManiaLinks

La balise *<manialinks>* est optionnelle et peut être ajoutée lorsqu'il est nécessaire de transmettre plusieurs ManiaLinks dans un fichier unique.

Exemple

Envoi de plusieurs ManiaLinks en une seule fois :

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialinks>
   <manialink version="1">
      <!-- Contenu du premier ManiaLink -->
   </manialink>
   <manialink version="1">
      <!-- Contenu du deuxième ManiaLink -->
   </manialink>
   <manialink version="1">
      <!-- Contenu du troisième ManiaLink -->
   /<manialink>
</manialinks>
```

ManiaLink

La balise <*manialink*> est le parrain de toutes les autres balises. Elle les entoure en tant que nœud racine de tous les autres éléments qui constituent un manialink. Et si un ManiaLink ne commence pas par <*manialink*>, il n'est tout simplement pas un ManiaLink.

Attributs

- **version="1"** – *Version du ManiaLink*
  Avec cet attribut, vous spécifiez la version du ManiaLink. En fonction de ce numéro de version, le ManiaLink pourra s'afficher de manière différente et certaines fonctionnalités seront ou non activées. Les ManiaLinks de Forever utilisent la version 0, valeur par défaut, alors que les ManiaLinks de ManiaPlanet utilisent la version 1. C'est cette valeur qui est recommandée pour le développement de nouveaux ManiaLinks. Consultez le paragraphe "Les ManiaLinks de version 1" dans le chapitre précédent.

- **background="0"** – *Visibilité de l'arrière-plan*
  Si cet attribut est mentionné et valué à 0, le ManiaLink n'aura pas d'arrire-plan spécifique et le jeu sera visible derrière le ManiaLink. C'est une sorte de fonctionnalité d'arrière-plan transparent.

Exemple

Un ManiaLink minimal :

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
   <!-- Contenu du ManiaLink -->
</manialink>
```

Timeout

Pour économiser du trafic, tous les ManiaLinks demandés sont mis en cache localement par ManiaPlanet, jusqu'à ce que le jeu soit arrêté. Cela permet, en cas d'appel d'un même ManiaLink une seconde fois, de n'utiliser que la copie locale au lieu de la redemander au serveur. Toutefois, ce qui peut être une fonctionnalité agréable pour les ManiaLinks statiques peut devenir un problème pour les dynamiques : les

dernières modifications ne s'affichent pas, ManiaPlanet ne rechargeant pas le fichier depuis le serveur.

Avec la balise <timeout>, vous pouvez désactiver cette mise en cache des fichiers ManiaLink : la valeur spécifiée dans le nœud indique le nombre de secondes avant que le jeu ne redemande le fichier au serveur. Typiquement, cette valeur est définie à 0 pour forcer ManiaPlanet à toujours recharger le fichier à partir du serveur.

Exemple

Squelette d'un manialink sans mise en cache :

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
   <timeout>0</timeout>
   <!-- Contenu du ManiaLink -->
</manialink>
```

Frame

Cet élément appelé cadre est destiné à regrouper plusieurs sous-éléments au sein d'un ensemble unique. Le cadre lui-même n'est pas visible, mais avec son facteur de position et de mise à l'échelle, il influence tous les éléments qu'il regroupe. Ainsi, si la position du cadre est modifiée, tous les éléments qu'il inclut sont déplacés vers les nouvelles coordonnées.

Les cadres doivent être utilisés lorsque plusieurs éléments du ManiaLink ont une finalité commune. Par exemple, si vous regroupez ensemble des éléments de menu dans un cadre, il suffit de déplacer sa position pour réorganiser le menu. Par contre, pour placer l'ensemble du menu vers un autre emplacement, il suffit de ne modifier que la position du cadre. En outre, vous pouvez inclure un cadre dans un autre, ce qui vous donne la possibilité de construire une structure hiérarchique dans votre ManiaLink.

Attributs

- **pos="x y z"** – *Position du cadre en p-system.*
- **posn="x y z"** – *Position du cadre en n-system.*
- **scale="factor"** – *Mise à l'échelle du cadre.*

- **id="identificateur"** – *Identificateur du cadre pour les ManiaScripts*
  Instance de la classe **`CGameManialinkFrame`**

Exemple

Utilisation d'un cadre pour combiner plusieurs Quads :

Code

```
<frame posn="50 50 0">
   <quad sizen="20 20" bgcolor="F00A" />
   <quad posn="-20 0 0" sizen="20 20" bgcolor="00FA" />
</frame>
```

Dans cet exemple, un cadre contient deux Quads positionnés côte-à-côte et s'harmonisant avec le cadre : si ce dernier est déplacé, les Quads resteront côte-à-côte à un autre emplacement sur l'écran.

Format

Avec la balise <*format*>, vous pouvez définir les valeurs par défaut de plusieurs attributs d'autres éléments. Les formats ainsi définis s'appliquent à tous les éléments du cadre dans lequel la balise <*format*> est placée, ainsi qu'à tous ses sous-cadres. En plaçant cette balise comme enfant direct dans la balise <*manialink*>, ces formats sont appliqués à tous les éléments du ManiaLink.

Les définitions spécifiées dans une balise <*format*> n'affectent que les valeurs par défaut des attributs des éléments <*quad*>, <*label*>, <*entry*> et <*fileentry*>, il reste possible de les spécifier dans chaque élément.

Attributs

- **bgcolor="RVBA"** – *Couleur de fond de l'élément.*
  Affecte les balises <*quad*>
  Les couleurs sont spécifiées comme un nombre hexadécimal à quatre chiffres, avec pour chaque une valeur comprise entre 0 et 9 et A à F. Les trois premiers chiffres représentent la valeur des canaux rouge, vert et bleu, et le dernier chiffre représente l'opacité, de 0 pour transparent à F pour opaque.
- **textsize="taille"** – *Taille du texte dans les éléments.*
  Affecte les balises <*label*>, <*entry*> et <*fileentry*>

- **textcolor="RVBA"** – *Couleur du texte de l'élément.*
  Affecte les balises <*label*>, <*entry*> et <*fileentry*>
- **focusareacolor1="RVBA"** – *Fond de l'élément désélectionné.*
  Affecte les balises <*label*>, <*entry*> et <*fileentry*>
- **focusareacolor2="RVBA"** – *Fond de l'élément sélectionné.*
  Affecte les balises <*label*>, <*entry*> et <*fileentry*>
  Les attributs *focusareacolor1* et *focusareacolor2* définissent la couleur de fond de l'élément. *focusareacolor1* définit la couleur à utiliser lorsque la souris n'est pas sur l'élément, et *focusareacolor2* définit la couleur lorsque la souris est placée sur l'élément.
- **style="NomDeStyle"** – Style de l'élément
  Affecte les balises <*label*>, <*entry*> et <*fileentry*>
  Avec l'attribut *style* vous pouvez utiliser les mêmes styles que ManiaPlanet utilise dans ses propres menus. Par exemple, avec *style="CardButtonSmall"*, vous pouvez créer le bouton noir qui devient blanc lorsque la souris est placée dessus. La liste des styles disponibles peut être visualisée sur le ManiaLink *styles*

Exemple

Utilisation de la balise <format> :

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1.0">
   <frame>
      <format bgcolor="F00A" />
      <quad sizen="10 10" />
      <quad posn="10 0" sizen="10 10" bgcolor="0F0A" />
   </frame>
   <quad posn="20 0" sizen="10 10" />
</manialink>
```

La balise <*format*> définit que la couleur d'arrière-plan sera par défaut en rouge. Cette couleur s'applique au premier Quad. Le deuxième Quad définit sa propre couleur d'arrière-plan, ce qui surcharge et remplace la valeur prédéfinie par <*format*>. Le troisième et dernier Quad n'est pas visible du tout, car il se trouve en dehors du nœud <*frame*> et donc la valeur prédéfinie par la balise <*format*> ne s'applique pas à lui.

Quad

Le Quad est l'un des plus importants éléments des ManiaLinks car il permet d'afficher des images et des formes. Il est même possible d'utiliser les modèles inclus dans ManiaPlanet, appelés *styles*, pour mieux intégrer votre ManiaLink dans le jeu. En outre, il est possible de lier un Quad à un autre ManiaLink ou à un site Web externe. Cette liaison est alors mise en surbrillance lorsque la souris est placée sur le Quad. En ce qui concerne les images, les Quads acceptent les formats .jpg, .png, .tga et .dds.

Attributs

- **pos="X Y Z"** – *Position du Quad en p-system.*
- **posn="X Y Z"** – *Position du Quad en n-system.*
- **size="largeur hauteur"** – *Taille du Quad en p-system.*
- **sizen="largeur hauteur"** – *Taille du Quad en n-system.*
- **scale="facteur"** – *Mise à l'échelle du Quad.*
- **halign="left|center|right"** – *Alignement horizontal du Quad*.
- **valign="top|center|bottom"** – *Alignement vertical du Quad*.
- **style="NomDeStyle"** – *Catégorie de style prédéfini.*
- **substyle="NomDeSousStyle"** – *Nom du style dans la catégorie*
  Les deux attributs *style* et *substyle* permettent de définir un type de mise en forme pour le quad, qui peut être une icône ou un fond avec une forme et une couleur prédéfinies. La liste complète des styles et sous-styles est disponible sur le ManiaLink "styles". Attention, ces attributs ne peuvent pas être utilisés en même temps que *image*/*imagefocus* et *bgcolor*.
- **image="FichierImage"** – *Image à afficher dans le Quad*.
- **imageid=" Identifiant"** – *Identifiant de l'image dans le dictionnaire*.
  Cet attribut permet de localiser l'image en récupérant son URL dans le dictionnaire (voir la balise <*dico*> plus loin). Les attributs *image* et *imageid* ne doivent pas être utilisés simultanément.
- **imagefocus="FichierImage"** – *Image à afficher dans le Quad alors que la souris est placée dessus.*
  Pour les deux attributs *image* et *imagefocus*, le nom du fichier image entre guillemets peut être soit l'URL complet vers une image (ex: http://www.imageprovider.com/12345.jpg"), soit une URL

relatif au fichier XML lui-même (ex: "./toto.png"). Attention, ces deux attributs ne peuvent pas être utilisés en même temps que *style/substyle* et que *bgcolor*.

- **imagefocusid=" Identifiant"** – *Identifiant de l'image dans le dictionnaire*.
  Cet attribut permet de localiser l'image en récupérant son URL dans le dictionnaire (voir la balise *<dico>* plus loin). Les attributs *imagefocus* et *imagefocusid* ne doivent pas être utilisés simultanément.
- **bgcolor="RGBA"** – *Couleur de fond du Quad*.
  RGBA représente la couleur, sous la forme d'un nombre hexadécimal, un chiffre hexa pour chaque composante de couleur et un pour l'opacité. Attention, cet attribut ne peut pas être utilisé en même temps que *style/substyle* et que *image/imagefocus*.
- **manialink="manialink"** – *Crée un lien vers un autre ManiaLink au moyen d'un URL ou d'un code*.
  Avec l'attibut *manialink*, le ManiaLink spécifié est ouvert et affiché avec ManiaPlanet. Le ManiaLink en question doit être un fichier XML ou un site web retournant un fichier XML valide.
- **manialinkid="Identifiant"** – *Identifiant du lien dans le dictionnaire*.
  Cet attribut permet de localiser le ManiaLink en récupérant son URL dans le dictionnaire (voir la balise *<dico>* plus loin). Les attributs *manialink* et *manialinkid* ne doivent pas être utilisés simultanément.
- **url="urlPage.html"** – *Crée un lien vers un site web*.
  Avec l'attribut *url*, le site web est ouvert dans le navigateur internet par défaut de l'utilisateur, et ManiaPlanet est réduit en icône.
- **urlid=" Identifiant"** – *Identifiant du lien dans le dictionnaire*.
  Cet attribut permet de localiser le ManiaLink en récupérant son URL dans le dictionnaire (voir la balise *<dico>* plus loin). Les attributs *url* et *urlid* ne doivent pas être utilisés simultanément.
- **id="identificateur"** – *Identificateur du Quad pour ManiaScript*.
  Instance de la classe **`CGameManialinkQuad`**.
- **scriptevents="1"** – *Active les évènements de souris pour le Quad.*

Exemple

Code

```
<quad sizen="50 10"
      style="Bgs1InRace"
      substyle="BgWindow1" />
<quad posn="0 -60" sizen="50 10"
      image="http://localhost/menu.png"
      imagefocus="http://localhost/menuHover.png"
      manialink="styles" />
```

Cet exemple affiche deux Quads, le premier contenant un style prédéfini de ManiaPlanet et le second une image personnalisée liée à un autre ManiaLink.

Label

En plus des Quads, les étiquettes (labels en anglais) sont les éléments les plus importants des ManiaLinks car elles permettent d'afficher n'importe quel type de texte, que ce soit sur une seule ou plusieurs lignes. Le texte à afficher est spécifié comme valeur de l'attribut *texte* ou être écrit entre la balise <*label*> et la balise </*label*>.

Une particularité intéressante des étiquettes est la possibilité d'avoir une traduction automatique de certains mots et groupes de mots, s'ils existent dans les fichiers de langue de ManiaPlanet. Ainsi, si vous écrivez juste le mot "Statistiques" dans un label, il sera traduit par "Statistik" pour les utilisateurs allemands de votre ManiaLink, par "Statistics" par les utilisateurs anglophones, etc. Si vous ne souhaitez pas que cette traduction automatique soit effectuée, ajoutez simplement un espace à la fin du texte.

Attributs

- **pos="X Y Z"** – *Position du label en p-system.*
- **posn="X Y Z"** – *Position du label en n-system.*
- **size="largeur hauteur"** – *Taille du label en p-system.*
- **sizen="largeur hauteur"** – *Taille du label en n-system.*
- **scale="facteur"** – *Mise à l'échelle du label.*
- **halign="left|center|right"** – *Alignement horizontal du label.*

- **valign="top|center|center2|bottom"** – *Alignement vertical*.
- **text="Texte"** – *Texte à afficher dans le label*.
  Vous pouvez utiliser les marqueurs de format dans le texte ($o, $w, $RVB, etc). De plus, comme mentionné plus haut, vous pouvez aussi spécifier le texte entre la balise <label> et la balise </label> au lieu d'utiliser cet attribut.
- **textid="Identifiant"** – *Identifiant du texte à afficher dans le label*.
  Spécifie l'identifiant du texte dans le dictionnaire (voir la balise <*dico*> plus loin). Les attributs *text* et *textid* ne doivent pas être utilisés simultanément.
- **style="NomDeStyle"** – *Catégorie de style prédéfini.*
  Il existe plusieurs styles prédéfinis s'appliquant aux balises label. Vous pouvez en spécifier un si vous le souhaitez afin de mettre en forme votre texte.
- **textsize="Nombre"** – *Taille du texte dans le label*.
- **textcolor="RVBA"** – *Couleur du texte dans le label.*
  Les attributs *style* et les attributs *textesize* et *textcolor* sont incompatibles entre eux puisque définissant la mise en forme du label. Dans la pratique, l'attribut *style* est prioritaire.
- **focusareacolor1="RVBA"** – *Fond de l'élément désélectionné*.
- **focusareacolor2="RVBA"** – *Fond de l'élément sélectionné.*
  Les attributs *focusareacolor1* et *focusareacolor2* définissent la couleur de fond à appliquer au label lorsque respectivement la souris n'est pas sur le label ou est au-dessus du label.
  Les attributs *style* et les attributs *focusareacolor1* et *focusareacolor2* sont incompatibles entre eux puisque définissant la mise en forme du label. L'attribut *style* est prioritaire.
- **autonewline="1"** – *Activation de la fonctionnalité d'insertion automatique de fins de ligne.*
  Si cet attribut est défini à *1*, les longues lignes seront automatiquement affichées sur plusieurs lignes afin de respecter la largeur du label. Sinon, seules les ruptures de lignes placées dans le texte lui-même seront respectées.
- **maxline="Nombre"** – *Limite du nombre de lignes.*
  Avec cet attribut, vous pouvez définir le nombre maximum de lignes pouvant être affichées dans le label. Les lignes au-delà de cette limite seront ignorées et non affichées.

- **manialink="manialink"** – *Crée un lien vers un autre ManiaLink au moyen d'un URL ou d'un code*.
  Avec l'attribut *manialink*, le ManiaLink spécifié est ouvert et affiché avec ManiaPlanet. Le ManiaLink en question doit être un fichier XML ou un site web retournant un fichier XML valide.
- **manialinkid="Identifiant"** – *Identifiant du lien dans le dictionnaire*.
  Cet attribut permet de localiser le ManiaLink en récupérant son URL dans le dictionnaire (voir la balise *<dico>* plus loin). Les attributs *manialink* et *manialinkid* ne doivent pas être utilisés simultanément.
- **url="urlPage.html"** – *Crée un lien vers un site web*.
  Avec l'attribut *url*, le site web est ouvert dans le navigateur internet par défaut de l'utilisateur, et ManiaPlanet est réduit en icône.
- **urlid=" Identifiant"** – *Identifiant du lien dans le dictionnaire*.
  Cet attribut permet de localiser le ManiaLink en récupérant son URL dans le dictionnaire (voir la balise *<dico>* plus loin). Les attributs *url* et *urlid* ne doivent pas être utilisés simultanément.
- **id="identificateur"** – *Identificateur du label pour ManiaScript*.
  Instance de la classe **`CGameManialinkLabel`**.
- **scriptevents="1"** – *Active les évènements de souris pour le label.*

Exemple

Code

```
<label style="CardButtonMediumWide"
       text="A button with a $F00red$g word" />
```

Un label avec style prédéfini et des marqueurs de mise en forme.

Audio

Comme son nom l'indique, cette balise permet d'ajouter une piste audio à votre ManiaLink. Cet élément est affiché comme simple bouton de démarrage et d'arrêt, avec lequel l'utilisateur peut arrêter ou lancer la lecture du fichier. Les formats de fichier .ogg et .mux sont acceptés.

Attributs

- **pos="X Y Z"** – *Position des boutons en p-system.*
- **posn="X Y Z"** – *Position des boutons en n-system.*
- **size="largeur hauteur"** – *Taille des boutons en p-system.*
- **sizen="largeur hauteur"** – *Taille des boutons en n-system.*
- **scale="facteur"** – *Mise à l'échelle des boutons.*
- **halign="left|center|right"** – *Alignement horizontal des boutons*.
- **valign="top|center|bottom"** – *Alignement vertical des boutons*.
- **data="fichier.ogg"** – *Nom du fichier audio à jouer.*
- **dataid="Identifiant"** – *Identifiant du fichier dans le dictionnaire*.
  Cet attribut permet de localiser le ManiaLink en recherchant l'URL du fichier audio dans le dictionnaire (voir la balise *<dico>* plus bas). Les attributs *data* et *dataid* ne doivent pas être utilisés simultanément.
- **play="1"** – *Démarrage automatique de la musique*.
  Si cet attribut est défini à *1*, la musique joue automatiquement dès que le ManiaLink a fini de s'afficher.
- **looping="0"** – *Désactivation du bouclage de l'audio*.
  Avec cet attribut défini à 0, la musique ne redémarre pas une fois jouée. Sinon, ManiaPlanet joue la musique en boucle.
- **id="identificateur"** – *Identificateur de la balise pour ManiaScript*.
  Instance de la classe **`CGameManialinkControl`**.
- **scriptevents="1"** – *Active les évènements de souris.*

Exemple

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1.0">
   <audio data="http://localhost/audio.ogg" play="1" />
   <quad posn="0 -20 1" sizen="20 10"
         halign="center" valign="center"
         image="http://localhost/welcome.dds"
         manialink="http://localhost/main.xml" />
</manialink>
```

Video

Cette balise est identique à la balise audio vue ci-dessus, si ce n'est qu'elle permet de jouer une vidéo dans votre ManiaLink, avec un bouton pour le contrôle de la lecture. La vidéo doit être au format .webm.

Attributs

- **pos="X Y Z"** – *Position de la vidéo en p-system.*
- **posn="X Y Z"** – *Position de la vidéo en n-system.*
- **size="largeur hauteur"** – *Taille de la vidéo en p-system.*
- **sizen="largeur hauteur"** – *Taille de la vidéo en n-system.*
- **scale="facteur"** – *Mise à l'échelle de la vidéo.*
- **halign="left|center|right"** – *Alignement horizontal de la vidéo*.
- **valign="top|center|bottom"** – *Alignement vertical de la vidéo*.
- **data="fichier.ogg"** – *Nom du fichier vidéo à jouer.*
- **dataid="Identifiant"** – *Identifiant du fichier dans le dictionnaire*.
  Cet attribut permet de localiser le ManiaLink en recherchant l'URL du fichier vidéo dans le dictionnaire (voir la balise <*dico*> plus bas). Les attributs *data* et *dataid* ne doivent pas être utilisés simultanément.
- **play="1"** – *Démarrage automatique de la vidéo*.
  Si cet attribut est défini à *1*, la vidéo démarre automatiquement dès que le ManiaLink a fini de s'afficher.
- **looping="0"** – *Désactivation du bouclage de la vidéo*.
  Avec cet attribut défini à 0, la vidéo ne redémarre pas une fois jouée. Sinon, ManiaPlanet joue la vidéo en boucle.
- **id="identificateur"** – *Identificateur de la balise pour ManiaScript*.
  Instance de la classe **`CGameManialinkMediaPlayer`**.
- **scriptevents="1"** – *Active les évènements de souris.*

Exemple

Code

```
<video data="http://localhost/welcome.webm" play="1" />
```

Music

La balise <*music*> est également utilisée pour lire des fichiers audio. Mais cette fois-ci sans contrôle de l'utilisateur sur la lecture du fichier. La balise <music> joue le fichier en arrière-plan et en boucle. Comme précédemment, seuls les fichiers .ogg et .mux sont acceptés. En tant que fonctionnalité supplémentaire, la musique n'arrête pas de jouer si vous modifiez ManiaLink, si la nouvelle utilise le même fichier audio dans sa balise <music>.

Note

> La balise <*music*> ne fonctionne que si elle est placée en tant que nœud fils de la balise <*manialink*>, c'est-à-dire en dehors de tout cadre.

Attributs

- **data="fichier.ogg"** – *Nom du fichier audio à jouer.*

Exemple

Exemple de la balise <music> avec son attribut :

Code

```
<music data="http://localhost/music.ogg" />
```

Include

La balise <*include*> permet d'inclure un autre fichier XML dans le ManiaLink actuel. Cela permet de globaliser des éléments fréquemment utilisés, comme un menu, dans un fichier séparé qu'il suffit ensuite d'inclure dans tous les ManiaLinks qui doivent être affichés. Ainsi, en cas de besoin de modification du menu, la simple modification du fichier inclus évite de devoir modifier tous les fichiers XML.

L'inclusion d'autres fichiers est limitée à un seul niveau. Ainsi, si vous pouvez inclure un fichier à partir du fichier "principal", vous ne pouvez pas inclure un autre fichier dans le fichier lui-même inclus afin d'éviter les inclusions circulaires : un fichier A incluant un fichier B incluant lui-même le fichier A.

Le fichier inclus doit être un fichier XML complet et valide, et non un ManiaLink complet. Cela signifie que le fichier inclus doit avoir un en-tête XML et un nœud racine unique (en général un <*frame*> qui contient les éléments réels à inclure) et ne doit pas avoir une balise <*manialink*>. Vous pouvez imaginer le processus inclus comme couper le ManiaLink principal en deux parties dont l'une se situe dans le fichier inclus. Avoir un <*manialink*> supplémentaire à ce point ne pourrait que cause des problèmes.

Note

N'oubliez pas que l'inclusion d'un fichier à l'aide d'une balise <*include*> augmente le temps de chargement du ManiaLink, car le fichier inclus doit être demandé au serveur séparément. Si vous avez un ManiaLink dynamique (c'est-à-dire basé sur PHP), vous devriez toujours envisager de préférer les instructions *include* et *require* de PHP à la place

Attributs

- **url="include.xml"** – *URL du fichier à inclure.*

Exemple

Fichier ManiaLink principal :

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1.0">
   <quad sizen="10 10" bgcolor="F00A" />
   <include url="file://Media/Manialinks/include.xml" />
</manialink>
```

Fichier include.xml inclus :

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<frame>
   <quad posn="10 0" sizen="10 10" bgcolor="0F0A" />
   <quad posn="20 0" sizen="10 10" bgcolor="00FA" />
</frame>
```

Le deuxième extrait de code montre le contenu fichier à inclure. Comme indiqué ci-dessus, il ne contient pas de balise <*manialink*>, mais contient une balise <*frame*> entourant les éléments réels à inclure afin que le fichier XML soit valide en n'ayant qu'un seul nœud racine.

Entry

Cette balise fait apparaître un champ de saisie et permet d'interagir avec l'utilisateur, lui permettant d'entrer n'importe quelle valeur. Bien sûr, une telle entrée n'a de sens que dans les ManiaLinks dynamiques, dans lesquels un script serveur (par exemple écrit en PHP) permet de récupérer et de traiter la valeur saisie. L'utilisation d'une balise <*entry*> est assez simple : vous spécifiez un nom unique, et à chaque endroit où ce nom apparaît dans un lien d'un label ou d'un quad, il est remplacé par la valeur actuelle de l'entrée. Vous devez vous rappeler qu'une entrée accepte toujours des valeurs de type chaîne de caractères, aussi votre script serveur doit valider la saisie effectuée dans le cas où des nombres sont attendus.

Attributs

- **pos="X Y Z"** – *Position de la zone de saisie en p-system.*
- **posn="X Y Z"** – *Position de la zone de saisie en n-system.*
- **size="largeur hauteur"** – *Taille de la zone de saisie en p-system.*
- **sizen="largeur hauteur"** – *Taille de la zone de saisie en n-system.*
- **scale="facteur"** – *Mise à l'échelle de la zone de saisie.*
- **halign="left|center|right"** – *Alignement horizontal de la zone de saisie*.
- **valign="top|center|bottom"** – *Alignement vertical de la zone de saisie*.
- **style="NomDeStyle "** – *Catégorie de style prédéfini.*
  Il existe plusieurs styles prédéfinis. Vous pouvez en spécifier un si vous le souhaitez afin de mettre en forme votre zone de saisie.
- **textsize="Nombre"** – *Taille du texte dans la zone de saisie*.
- **textcolor="RVBA"** – *Couleur du texte dans la zone de saisie.*
  Les attributs *style* et les attributs *textesize* et *textcolor* sont incompatibles entre eux puisque définissant la mise en forme de la zone de saisie. Dans la pratique, l'attribut *style* est prioritaire.

- **focusareacolor1="RVBA"** – *Fond de l'élément désélectionné*.
- **focusareacolor2="RVBA"** – *Fond de l'élément sélectionné.*
  Les attributs *focusareacolor1* et *focusareacolor2* définissent la couleur de fond à appliquer à la zone de saisie lorsque la souris est respectivement en-dehors et au-dessus de cette dernière.
  Les attributs *style* et les attributs *focusareacolor1* et *focusareacolor2* sont incompatibles entre eux puisque définissant la mise en forme de la zone de saisie. Dans la pratique, l'attribut *style* est prioritaire.
- **name="NomEntree"** – *Nom unique de la zone de saisie.*
  Comme mentionné préalablement, cet attribut est important : à tous les endroits où ce nom apparaît dans les liens, il est remplacé par la valeur entrée par l'utilisateur.
- **default="Valeur"** – *Valeur par défaut de la zone de saisie.*
  Lors du chargement du ManiaLink, la zone de saisie est initialisée avec la valeur de cet attribut, qui reste telle quelle tant que l'utilisateur ne l'a pas modifié.
- **autonewline="1"** – *Activation de la fonctionnalité d'insertion automatique de fins de ligne.*
  Si cet attribut est défini à *1*, les longues lignes seront automatiquement affichées sur plusieurs lignes afin de respecter la largeur de la zone de saisie.
- **id="identificateur"** – *Identificateur du label pour ManiaScript*.
  Instance de la classe **`CGameManialinkEntry`**.
- **scriptevents="1"** – *Active les évènements de souris pour l'objet.*

Exemple

Petit exemple de ManiaLink avec une zone de saisie et un lien :

Code

```
<entry sizen="50 5"
       style="TextValueSmall"
       name="inputValue"
       default="Hello World!" />
<label posn="0 -10"
       style="CardButtonMedium"
       url="http://localhost/script.php?value=inputValue"
       text="Send Value" />
```

Cet exemple contient une zone de saisie nommée *inputValue* et une étiquette qui permet d'envoyer la valeur saisie à un script PHP sous forme d'un paramètre. La valeur du paramètre *value* sera remplacée par la valeur actuelle de l'entrée lorsque l'utilisateur cliquera sur le label. Reportez-vous au paragraphe "Un exemple complexe", page 60, pour plus d'informations.

FileEntry

La balise <*fileentry*> fonctionne de la même manière que la balise <*entry*>, avec une différence de taille : l'utilisateur ne saisit pas de valeur, mais peut sélectionner un fichier à téléverser sur le serveur. Comme la balise <*entry*>, la balise <*fileentry*> utilise un nom unique, qui n'a de sens que sur dans les ManiaLinks dynamiques pour lesquels un script serveur peut gérer le fichier téléversé.

Lorsque l'utilisateur clique sur la zone <*fileentry*>, une boîte de dialogue de sélection de fichier apparait, et l'utilisateur peut choisir un fichier à partir de son ordinateur. Ce fichier est alors transmis à l'aide d'une requête POST. Reportez-vous au paragraphe "Un exemple complexe", page 60, pour plus d'informations.

Note

> Le script serveur doit toujours vérifier le fichier téléversé par l'utilisateur avant de l'enregistrer. Sauvegarder des fichiers sans les vérifier est un énorme risque pour la sécurité !

Attributs

- **pos="X Y Z"** – *Position de l'objet en p-system.*
- **posn="X Y Z"** – *Position de l'objet en n-system.*
- **size="largeur hauteur"** – *Taille de l'objet en p-system.*
- **sizen="largeur hauteur"** – *Taille de l'objet en n-system.*
- **scale="facteur"** – *Mise à l'échelle de l'objet.*
- **halign="left|center|right"** – *Alignement horizontal de l'objet.*
- **valign="top|center|bottom"** – *Alignement vertical de l'objet.*

- **style="NomDeStyle "** – *Catégorie de style prédéfini.*
  Il existe plusieurs styles prédéfinis. Vous pouvez en spécifier un si vous le souhaitez afin de mettre en forme votre objet.
- **textsize="Nombre"** – *Taille du texte dans l'objet*.
- **textcolor="RVBA"** – *Couleur du texte dans l'objet.*
  Les attributs *style* et les attributs *textesize* et *textcolor* sont incompatibles entre eux puisque définissant la mise en forme de l'objet. Dans la pratique, l'attribut *style* est prioritaire.
- **focusareacolor1="RVBA"** – *Fond de l'élément désélectionné*.
- **focusareacolor2="RVBA"** – *Fond de l'élément sélectionné.*
  Les attributs *focusareacolor1* et *focusareacolor2* définissent la couleur de fond à appliquer à l'objet lorsque respectivement la souris n'est pas sur l'objet ou est au-dessus de l'objet.
  Les attributs *style* et les attributs *focusareacolor1* et *focusareacolor2* sont incompatibles entre eux puisque définissant la mise en forme de l'objet. Dans la pratique, l'attribut *style* est prioritaire.
- **name="NomEntree"** – *Nom unique de l'objet.*
  Comme pour les zones de saisie, à tous les endroits où ce nom apparaît dans les liens, il est remplacé par le nom du fichier sélectionné.
- **folder="Chemin"** – *Dossier source*
  Avec cet attribut, vous pouvez aider l'utilisateur à trouver le fichier en y spécifiant le répertoire de base que l'utilisateur peut parcourir. Ce dossier est toujours relatif au dossier ManiaPlanet du répertoire "Mes Documents". L'utilisateur ne peut parcourir les dossiers supérieurs. Ainsi, par exemple, si vous souhaitez que l'utilisateur puisse téléverser des pistes, vous pouvez spécifier "Maps" dans cet attribut. Gardez cependant à l'esprit que l'utilisateur est toujours en mesure de télécharger tout fichier qu'il souhaite : une double vérification est nécessaire sur le serveur !
- **default="Valeur"** – *Valeur par défaut de l'objet.*
  Lors du chargement du ManiaLink, l'objet est initialisé avec la valeur de cet attribut, qui reste telle quelle tant que l'utilisateur n'a pas choisi de fichier sur son disque.
- **id="identificateur"** – *Identificateur du label pour ManiaScript*.
  Instance de la classe **`CGameManialinkFileEntry`**.
- **scriptevents="1"** – *Active les évènements de souris pour l'objet.*

- **autonewline="1"** – *Activation de la fonctionnalité d'insertion automatique de fins de ligne.*
  Si cet attribut est défini à *1*, les longues lignes seront automatiquement affichées sur plusieurs lignes afin de respecter la largeur de l'objet.

Exemple

Une balise FileEntry et bouton permettant d'envoyer le fichier :

Code

```
<fileentry sizen="50 5"
           style="TextValueSmall"
           name="F"
           folder="Maps"
           default="Choisissez une piste..."
/>
<label posn="0 -10"
       style="CardButtonMedium"
       text="Send File"
       url="POST(http://localhost/script.php?file=F,F)"
/>
```

Cet exemple est similaire à celui pour la balise *<entry>*, avec une différence : l'utilisation la méthode POST () pour l'envoi du résultat à un autre ManiaLink ou à un site Web. Encore une fois, reportez-vous au paragraphe "Un exemple complexe", page 60, pour plus d'informations.

Dico

Avec la balise *<dico>* et un certain nombre de balises *<language>*, vous pouvez traduire votre ManiaLink dans d'autres langues, ce qui permettra aux joueurs anglophones de visualiser la version anglaise, alors qu'un joueur germanophone accèdera à la version allemande. La langue d'affichage sera sélectionnée par le jeu en fonction de la langue du joueur (qu'il a sélectionnée dans la configuration de ManiaPlanet). Si elle est disponible, elle sera utilisée, sinon ManiaPlanet utilisera la version anglaise. Et si cette dernière n'est pas disponible, rien ne sera affiché.

Rendre un ManiaLink multilingue est très simple : au lieu de spécifier les termes exacts dans les balises *<label>*, n'utilisez que des identifiants :

Exemple

Un ManiaLink simple mais multilingue :

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
   <timeout>0</timeout>
   <!-- Contenu du ManiaLink -->
   <label posn="-20 0" textid="example" />
   <label posn="-20 -10" textid="thanks" />
   <quad sizen="10 5" imageid="img" />

   <!-- Traductions des IDs -->
   <dico>
      <language id="en">
         <example>Example</example>
         <thanks>Thanks Nadeo for the ManiaLinks!</thanks>
         <img>http://localhost/manialink/en.png</img>
      </language>
      <language id="fr">
         <example>Exemple</example>
         <thanks>Merci Nadeo pour les ManiaLinks!</thanks>
         <img>http://localhost/manialink/fr.png</img>
      </language>
      <language id="de">
         <example>Beispiel</example>
         <thanks>Danke Nadeo für die ManiaLinks!</thanks>
         <img>http://localhost/manialink/de.png</img>
      </language>
   </dico>
</manialink>
```

La première partie du ManiaLink contient les éléments à afficher à l'utilisateur, mais dans lesquels les attributs *text* et *image* ont été remplacés par des attributs *textid* et *imageid*. Ceci indique à ManiaPlanet qu'il doit rechercher l'identifiant spécifié dans la balise <*dico*> et afficher le texte correspondant pour la langue de l'utilisateur.

Ces identifiants doivent commencer par une lettre, ne contenir que des lettres, des chiffres et des caractères de soulignement "_".

Liste des attributs utilisant le dictionnaire :

- **<quad>**..... *imageid*, *imagefocusid*, *urlid*, *manialinkid*
- **<label>**..... *textid*, *urlid*, *manialinkid*
- **<audio> ....** *dataid*
- **<video> ....** *dataid*

La deuxième partie de l'exemple montre la traduction réelle des identifiants, entre la balise <*dico*> et la balise </*dico*>. Pour chaque langue à prendre en charge, on ajoute une balise <*language*>, contenant l'attrtibut *id* dont la valeur est l'indicatif de la langue prise en charge par ses nœuds enfants, parmi la liste suivante :

id	Langue	id	Langue
cz	Tchèque	nb	Norvégien
da	Danois	nl	Néerlandais
de	Allemand	pl	Polonais
en	Anglais	pt	Portugais
es	Espagnol	pt_BR	Brésilien
fr	Français	ro	Roumain
hu	Hongrois	ru	Russe
it	Italien	sk	Slovaque
jp	Japonais	tr	Turc
kr	Coréen	zh	Chinois

Pour chaque langue, entre la balise <*language*> et la balise </*language*>, on ajoute, sous forme de balises, les identifiants nommés dans les balises <*quad*>, <*label*>, <*audio*> et <*video*> du ManiaLink.

Entre la balise de début de l'identifiant et la balise de fin correspondante, on placera le texte localisé correspondant, qui cette fois pourra contenir toutes sortes de caractères accentués ou non (sauf le caractères ayant une signification XML que l'on remplacera par les entités HTML correspondantes, comme indiqué dans le paragraphe concernant les caractères spéciaux en page 15.

Script

La balise <*script*> est peut-être la balise la plus simple, mais elle est l'élément le plus puissant d'un ManiaLink : avec cette balise, vous pouvez inclure un ManiaScript dans votre ManiaLink. Cependant, comme certains éléments du ManiaScript peuvent invalider la syntaxe XML, il est recommandé de placer le texte du ManiaScript dans un commentaire XML.

Il est possible d'utiliser plusieurs balises <*script*> dans votre ManiaLink. Ces éléments seront rattachés lors du chargement du ManiaLink et exécutées comme un grand ManiaScript unique. Il n'y a donc pas de différences entre trois ManiaScript et un seul grand ManiaLink, inséré par exemple en fin de ManiaLink.

Pour plus d'informations, consultez le chapitre concernant le ManiaScript dans les ManiaLinks.

Exemple

Un simple exemple de ManiaLink contenant du ManiaScript :

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
   <quad id="myQuad" posn="10 0" sizen="20 20"
         bgcolor="F00A" ScriptEvents="1" />
   <script><!--
while(True) {
   foreach(Event in PendingEvents) {
      if (Event.Type ==
          CGameManialinkScriptEvent::Type::MouseClick) {
         Page.MainFrame.Controls["myQuad"].PosnX *= -1;
      }
   }
   yield;
}
--></script>
</manialink>
```

À ce stade, je ne vais pas expliquer en détail le ManiaScript. Enregistrez ce code en tant que nouveau fichier XML, ouvrez-le dans le navigateur ManiaPlanet et cliquez sur le Quad.

Le ManiaScript
Le langage de manipulation des ManiaLinks

ManiaScript est un langage indépendant développé par Nadeo pour rendre les différentes parties du jeu ManiaPlanet plus dynamiques. Cela s'applique également à ManiaLinks : avec ManiaPlanet, vous avez la possibilité de manipuler et de modifier les ManiaLinks après leur affichage et de réagir aux actions des utilisateurs, comme le mouvement de la souris ou les entrées du clavier. ManiaScript est en quelque sorte aux ManiaLinks ce que JavaScript est aux pages HTML.

Ce didacticiel n'a pas pour but d'expliquer les principes de base de la syntaxe des ManiaScripts, donc si vous ne connaissez pas encore le langage, vous devrez faire appel à un autre didacticiel.

Ce qui suit requiert des connaissances de base concernant ManiaScript.

Accès aux éléments d'un ManiaLink

L'un des points fondamentaux de ManiaScript est la manipulation des éléments des ManiaLinks, étape nécessaire afin de pouvoir modifier leurs attributs. Nous devons obtenir l'instance de l'objet, qui représente l'élément ManiaLink, et pour cela il existe deux façons différentes utilisant la variable globale *Page* :

Page.MainFrame.Controls[*IdElement*] – *Accès à un élément au travers du cadre principal*

La première façon d'accéder à un élément de ManiaLink est d'y faire référence par rapport au cadre principal. La propriété **`MainFrame`** de type **`CGameManialinkFrame`** représente la balise <*maniaLink*> du ManiaLink, et comme le type l'indique, il est considéré comme un cadre ou une balise <*frame*>. Ce cadre publie une propriété **`Controls`**, qui est un tableau associatif des identificateurs des objets associés aux éléments

existants dans le ManiaLink (de type **CGameManialinkControl**, voir ci-dessous). Ces identificateurs sont les valeurs données aux attributs *id* des éléments, ce qui implique que les éléments de ManiaLink auxquels vous souhaitez accéder doivent avoir des *id* distincts.

La propriété **Controls** étant un tableau, vous pouvez également y accéder par leur index. Ainsi, **Page.MainFrame.Controls[3]** vous retourne le 4ème nœud enfant de la balise <*manialink*> (le premier élément étant à l'index 0). La spécification d'identificateur est cependant recommandée, car en cas de changement d'ordre des éléments ou en cas d'ajout d'autres éléments, les index peuvent être modifiés.

Note

> La propriété **Controls** de l'objet **Page.MainFrame** ne contient que les nœuds enfants directs du cadre (c'est-à-dire tous les enfants directs de <*manialink*>). Cela signifie que si votre manialink contient un élément <*frame*>, vous n'aurez pas directement accès aux éléments qui le composent à moins de considérer l'élément de **Controls** correspondant au cadre comme une instance de la classe **CGameManialinkFrame** (transtypage), puis de lire la valeur de la propriété **Controls** de cet élément, voir l'exemple ci-dessous.

Page.GetFirstChild(*IdElement*) – *Recherche de l'élément*

La seconde technique d'accès à un élément de ManiaLink est d'utiliser la méthode **GetFirstChild** de l'objet **Page** en spécifiant en paramètre l'identificateur de l'élément souhaité. L'avantage de cette technique sur la précédente est que la recherche s'effectue sur l'ensemble des nœuds enfants du ManiaLink, quelque soit leur niveau. Cette méthode retourne indifféremment les enfants directs du ManiaLink comme les éléments situés à plusieurs niveaux de <*frame*> enfants.

Vous pouvez également utiliser cette méthode sur un objet <*frame*> pour ne rechercher que parmi ses nœuds descendants car les méthodes **Page.GetFirstChild()** et **Page.MainFrame.GetFirstChild()** sont équivalentes.

Exemple

Après cet exposé, un petit exemple devrait aider à tout expliquer, en particulier le problème avec les cadres imbriqués (cf note ci-dessus) :

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
  <timeout>0</timeout>
  <quad id="exQuad" />
  <frame id="outFrame">
    <label id="exLabel" />
    <frame id="inFrame">
      <entry id="exEntry" />
    </frame>
  </frame>
  <script><!--
declare CGameManialinkQuad quad;
declare CGameManialinkLabel label;
declare CGameManialinkEntry entry;
declare CGameManialinkFrame frame;

// Access elements over Controls Array
quad = (Page.MainFrame.Controls["exQuad"]
        as CGameManialinkQuad);

// We cannot access exLabel and exEntry, so...
frame = (Page.MainFrame.Controls["outFrame"]
         as CGameManialinkFrame);
label = (frame.Controls["exLabel"]
         as CGameManialinkLabel);

// Now we set frame to the inFrame element, for exEntry
frame = (frame.Controls["inFrame"]
         as CGameManialinkFrame);
entry = (frame.Controls["exEntry"]
         as CGameManialinkEntry);

// The same elements accessed via GetFirstChild()
quad = (Page.GetFirstChild("exQuad")
        as CGameManialinkQuad);
label = (Page.GetFirstChild("exLabel")
         as CGameManialinkLabel);
entry = (Page.GetFirstChild("exEntry")
         as CGameManialinkEntry);
--></script>
</manialink>
```

Quel est alors l'avantage de la propriété **`Controls`**, si la méthode **`GetFirstChild()`** semble être beaucoup plus simple ? La disponibilité

complète de l'arborescence des éléments : on y voit toujours que l'objet *inFrame* est à l'intérieur de l'objet *outFrame*, et cela peut s'avérer utile : imaginez que vous avez développé une liste contenant des choix placés dans une balise <*frame*>. Au lieu d'avoir à donner des noms distincts à chaque choix pour y accéder au moyen de la méthode **`GetFirstChild()`**, vous pouvez accéder à cet ensemble au moyen d'une simple boucle sur la propriété **`Controls`**.

Un autre point déroutant de l'exemple ci-dessus est le transtypage (*cast*) systématique des éléments. La raison est que la propriété **`Controls`** et la méthode **`GetFirstChild()`** renvoient toujours un objet de la classe **`CGameManialinkControl`**. Qui est la classe de base de celles dont font partie tous les éléments des ManiaLink qui sont des classes héritées et qui spécialisent les fonctionnalités de la classe parente. Aussi, afin de pouvoir accéder aux membres de la classe héritée faut-il faire un transtypage.

Les classes spécifiques aux ManiaLinks

Pour manipuler les éléments d'un ManiaLink, ManiaScript fournit un ensemble de classes. La classe de base, qui est toujours renvoyée par la propriété **`Controls`** et la méthode **`GetFirstChild()`**, est la classe **`CGameManialinkControl`**. Cette dernière fournit des propriétés et des méthodes permettant de modifier la position et la visibilité de l'élément. Toutes les autres classes sont dérivées de cette classe de base et étendent les possibilités. Par exemple, la classe **`CGameManialinkLabel`** publie une méthode **`SetText()`** dont le rôle est de définir le texte affiché. La liste suivante détaille les classes disponibles pour les éléments de ManiaLinks, incluant une description de l'élément qu'elles représentent et leur arbre d'héritage.

Chaque propriété et méthode sera spécifiée sous la forme :

- Type de la propriété ou type de la valeur de retour de la méthode en italiques, suivi de :
- Nom de la propriété ou de la méthode en gras, suivi de :
- Pour une méthode uniquement : liste des paramètres entre parenthèses, suivi de :
- À la ligne suivante, description de la propriété ou de la méthode, suivi de :
- Pour les méthodes, description des paramètres.

CGameManialinkControl

La classe de base.

Propriétés :

- *Text* **Id**
  Identificateur de l'objet.
- *Real* **PosnX**
  Position X de l'objet en n-system.
- *Real* **PosnY**
  Position Y de l'objet en n-system.
- *Real* **PosnZ**
  Position Z de l'objet en n-system.

Méthodes :

- *Void* **Hide()**
  Masque l'objet.
- *Void* **Show()**
  Affiche l'objet s'il a été caché.
- *Void* **Unload()**
  Supprime totalement l'objet du ManiaLink. Une fois cette méthode exécutée, l'objet n'existe plus, toute tentative pour y accéder se solde par une erreur.

CGameManialinkFrame

Hérite de CGameManialinkControl

Représente les objets <*frame*>

Propriétés :

- *Array CGameManialinkControl[Text]* **Controls**
  Objets contenus par la frame.

Méthodes :

- *CGameManialinkControl* **GetFirstChild(Text)**
  Retourne le premier objet inclus dans la frame ayant cet ID.
  Paramètres :
  - **Text** : Identificateur de l'objet à retourner.

CGameManialinkQuad

Hérite de CGameManialinkControl

Représente les objets <*quad*>

Méthodes :

- *Void* **ChangeImageUrl(Text)**
  Modifie l'image affichée par le quad en spécifiant une nouvelle URL.
  Paramètres :
  - **Text** : URL de l'image à afficher dans le quad.

  Note : pour que cette méthode fonctionne, il faut que le quad ait déjà une image affichée au moyen de son attribut *image* et que l'URL spécifié par le paramètre soit un URL complet.

CGameManialinkLabel

Hérite de CGameManialinkControl

Représente les objets <*label*>

Méthodes :

- *Void* **SetText(Text)**
  Modifie le texte affiché dans l'objet.
  Paramètres :
  - **Text** : Nouveau texte à afficher dans le label.

CGameManialinkEntry

Hérite de CGameManialinkControl

Représente les objets <*entry*>

Propriétés :

- *Text* **Value**
  La valeur actuelle de la zone de saisie.

CGameManialinkFileEntry

Hérite de CGameManialinkControl

Représente les objets <*fileentry*>

Propriétés :

- *Text* **FullFileName** (lecture seule)
  Retourne le chemin complet et le nom du fichier sélectionné dans l'objet fileentry.

Note

> Une autre balise, qui ne dispose pas de sa propre classe, est <*audio*> : cette balise est directement représentée par **`CGameManialinkControl`** et il n'est donc possible que de modifier sa position et sa visibilité. La balise <*manialink*> est une instance de la classe **`CGameManialinkFrame`** et est toujours accessible via **`Page.MainFrame`**. Les balises non positionnables comme <*music*> ou <*format*> ne sont pas accessibles via un ID, car modifier par exemple leur position n'aurait aucun sens.

Gestion des événements

Maintenant que nous savons comment manipuler certains éléments du ManiaLink, nous devons évidemment savoir comment réagir sur les entrées de l'utilisateur, comme le mouvement de la souris, les clics et les pressions de touches. Pour cela, ManiaScript fournit différents types d'événements :

- **MouseClick**
  L'utilisateur a cliqué avec le bouton principal de la souris sur l'élément.
- **MouseOver**
  L'utilisateur a placé la souris au-dessus de l'élément.
- **MouseOut**
  L'utilisateur a placé la souris en-dehors de l'élément.
- **KeyPress**
  L'utilisateur a pressé une touche du clavier.

Il existe cependant quelques limitations concernant les événements, ce qui peut entraver le fonctionnement du ManiaScript :

- Seuls les événements qui ne sont pas gérés par l'élément lui-même peuvent être gérés dans un ManiaScript. Cela signifie que, par exemple, Une zone de saisie ne créera jamais d'événements *KeyPress*, car elle les gère elle-même en insérant le caractère

pressé dans le champ. De même, un quad ou un label liés ne créeront jamais d'événements de souris.

- Les événements de souris (*MouseClick*, *MouseOver* et *MouseOut*) nécessitent la définition de l'attribut *ScriptEvents* sur 1 pour les éléments correspondants.
- Chaque événement n'est déclenché qu'une seule fois, l'ordre Z détermine l'élément qui l'obtient. Par exemple, un évènement *MouseClick* ne sera reçu que par l'élément qui est au-dessus de tous les autres, c'est-à-dire celui dont la valeur Z est la plus élevée en cas de positionnement n-system ou de valeur Z la plus basse en cas de positionnement p-system. Tous les autres éléments derrière ce premier ne recevront pas la notification du *MouseClick*.

Jetons un coup d'œil sur une boucle de gestion d'événements typique d'un ManiaScript :

Code

```
main() {
  // #1 - À exécuter à l'initialisation
  while(True) {
    // #2 - À toujours exécuter
    foreach(Event in PendingEvents) {
      switch(Event.Type) {
        case CGameManialinkScriptEvent::Type::MouseClick: {
          // #3 - À exécuter sur MouseClick
        }
        case CGameManialinkScriptEvent::Type::MouseOver: {
          // #4 - À exécuter sur MouseOver
        }
        case CGameManialinkScriptEvent::Type::MouseOut: {
          // #5 - À exécuter sur MouseOut
        }
        case CGameManialinkScriptEvent::Type::KeyPress: {
          // #6 - À exécuter sur KeyPress
        }
      }
    }
    yield;
  }
}
```

Fondamentalement, nous avons six emplacements marqués par des commentaires (#1, #2, … #6), où nous pouvons ajouter du code

supplémentaire pour gérer les différentes situations. Si certaines ne sont pas requises, vous pouvez supprimer le commentaire et le code qui l'entoure (par exemple le *case xxxx { }* correspondant).

Si vous souhaitez exécuter du code une seule fois juste après le chargement du ManiaLink, par exemple pour masquer certains éléments qui devraient être affichés plus tard, vous pouvez le faire en dehors de la boucle sans fin, à l'emplacement du commentaire #1.

Le code à la position #2 sera toujours exécuté, en boucle. Ceci est le plus adapté aux animations puisque basées sur le temps qui passe, car plus souvent ces instructions sont exécutées, plus elles sont adaptées.

La boucle *foreach* est exécutée pour chaque événement réel déclenché par l'un des éléments. Vous pouvez utiliser une instruction *switch* pour les distinguer, comme indiqué dans l'exemple avec les positions #3 à #6.

Ce code est modifiable par vos soins comme vous l'entendez puisqu'au final, les seules instructions obligatoires sont la boucle infinie, la boucle *foreach* pour gérer les événements entrants et le *yield* avant la fin de la boucle infinie.

Un exemple complexe

Pour rendre tout cela un peu plus clair, je souhaite vous donner un exemple complet qui démontre l'utilisation d'un ManiaScript dans un ManiaLink.

Si cet exemple reste très basique, il montre cependant comment gérer les événements sans interférer avec une boucle d'animation.

Si vous avez des problèmes de compréhension de l'exemple, vous pouvez simplement copier et coller le code XML dans un fichier, le modifier un peu et voir comment ces changements se traduisent.

Notre exemple a pour but d'afficher une animation continue dans un *quad* se déplaçant en cercle.

De plus, ce *quad* doit réagir aux événements souris de l'utilisateur : lors de la réception d'un *MouseClick*, *MouseOver* ou *MouseOut*, le *quad* doit changer de couleur pour visualiser l'événement reçu.

Jetons un coup d'œil sur le code XML lui-même:

Code

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1" background="0">
  <timeout>0</timeout>
  <!-- Container pour changer les positions d'un coup -->
  <frame id="container">
    <!-- Quad qui recevra les événements de souris -->
    <quad id="cible" posn="0 0 5" sizen="20 20"
          ScriptEvents="1" />
    <!-- Comme on ne peut changer l'attribut bgcolor d'un
         quad, on utilise plusieurs quads et on n'en
         n'affiche qu'un -->
    <quad id="rouge" sizen="20 20" bgcolor="A00A" />
    <quad id="vert" sizen="20 20" bgcolor="080A" />
    <quad id="bleu" sizen="20 20" bgcolor="00AA" />
  </frame>
  <script><!-
#Include "MathLib" as MathLib

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Affiche seulement le quad ayant l'ID spécifié
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Void AffichageQuad(Text ControlID) {
  declare CGameManialinkFrame Container <=>
          (Page.MainFrame.Controls["container"]
          as CGameManialinkFrame);
  foreach(Quad in Container.Controls) {
    if (Quad.Id == "cible" || Quad.Id == ControlID) {
      Quad.Show();
    } else {
      Quad.Hide();
    }
  }
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Initialisation de l'animation
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Void Initialisation() {
  // Sauver l'heure d'initialisation
  declare Integer Debut for Page = CurrentTime;
  // Afficher le quad rouge par défaut
  AfficheQuad("rouge");
}
```

Code

```
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Calcul l'état de l'animation et déplace le container
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Void Animation() {
  // Récupérer la variable StartTime
  declare Integer Debut for Page;
  // Calcul de la difference de temps en secondes
  declare Real DiffT = (CurrentTime - Debut) / 1000.;
  // Met à jour la position du container pour l'animer
  declare CGameManialinkControl Container <=>
          Page.MainFrame.Controls["container"];
  Container.PosnX = 30 * MathLib::Sin(TimeDiff);
  Container.PosnY = 30 * MathLib::Cos(TimeDiff);
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Fonction principale
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

main() {
  Initialisation();
  while(True) {
    Animation();
    foreach(Event in PendingEvents) {
      switch(Event.Type) {
        case CGameManialinkScriptEvent::Type::MouseClick: {
          AffichageQuad("bleu");
        }
        case CGameManialinkScriptEvent::Type::MouseOver: {
          AffichageQuad("vert");
        }
        case CGameManialinkScriptEvent::Type::MouseOut: {
          AffichageQuad("rouge");
        }
      }
    }
    yield;
  }
}
--></script>
</manialink>
```

Le premier problème à résoudre est que nous ne pouvons pas modifier le *bgcolor* d'un Quad dans un ManiaScript. Donc nous utilisons plusieurs

quads avec la même taille et la même position, mais avec des couleurs différentes, et nous ne montrons que celui qui a la couleur voulue. Ces quads ont les identifiants *rouge*, *vert* et *bleu*.

Afin d'éviter la surveillance des évènements de souris pour chacun des trois quads, un quatrième quad a été ajouté, nommé *cible*, de même taille et de même position, mais placé en avant des autres quads dans le but d'intercepter les événements de souris, et sans aucune couleur de fond pour qu'il reste invisible. Ce quad est le seul élément ayant l'attribut *ScriptEvents* à *1*.

Comme tous ces quads doivent toujours avoir la même position, nous les plaçons tous dans une balise *frame* nommée container : c'est cette balise que l'on déplace pour animer les quads.

Examinons le ManiaScript, qui est l'élément important de notre exemple. Ce dernier comporte trois fonctions : **`AffichageQuad(ControlID)`** dont le rôle est de n'afficher que l'un des quads potentiellement visibles, en veillant à ce que le quad *cible* soit toujours apte à capturer les événements de souris. Par exemple, **`AffichageQuad("bleu")`** rend visible le quad de même nom tout en masquant les autres quads *sauf* celui nommé *cible*. **`Initialisation()`** est appelé une fois lorsque le ManiaLink finit de se charger pour initialiser toutes les valeurs dont nous avons besoin, en l'occurrence sauvegarder l'heure courante et afficher le quad *rouge*. **`Animation()`** pour sa part calcule et met à jour la position actuelle des quads pour réaliser le mouvement circulaire, en fonction de l'intervalle de temps par rapport à l'heure mémorisée préalablement. Cette fonction est appelée à chaque fois qu'aucune autre tâche ne monopolise le script afin que l'animation soit aussi fluide que possible.

Enfin, la fonction **`main()`** du ManiaScript implémente le comportement concret "autour" de nos fonctions. En première ligne, l'appel à la fonction **`Initialisation()`** permet de préparer notre code avant d'entrer dans la boucle infinie, elle-même divisée en trois parties :

- Mise à jour de la position des quads grâce à l'appel de la fonction **`Animation()`**,
- Prise en charge des nouveaux événements de souris. Selon le type de l'événement à traiter, nous montrons un quad de couleur différente, afin d'observer les événements qui ont été pris en compte. Ainsi le quad devient bleu lorsque vous cliquez dessus,

mais devient rouge une fois que la souris a été déplacée en dehors de celui-ci.

- L'appel à *yield* rend le contrôle à ManiaPlanet, afin que le jeu puisse de nouveau entre autres remplir le tableau *PendingEvents* avec les nouveaux événements déclenchés.

Dans cet exemple, un type d'événement n'est pas géré : *KeyPress*. Vous pouvez à titre d'exercice modifier le code du ManiaScript et du ManiaLink afin de le prendre en compte et modifier la couleur du quad en jaune si l'utilisateur appuie sur n'importe quelle touche.

Trucs et astuces
Morceaux choisis qui gagnent à être connus

Le dernier chapitre de ce didacticiel traite d'un ensemble de sujets particuliers ne correspondant pas aux chapitres précédents, ou trop importants pour les y décrire. Les paragraphes qui suivent ne sont pas liés les uns aux autres, de sorte que vous pouvez ne lire que les parties qui vous intéressent.

Exemple de saisie utilisateur et de téléversement

Une bonne connaissance du PHP est requise.

Avec des objets *entry* et *FileEntry*, l'utilisateur a la possibilité d'envoyer des informations et des fichiers au ManiaLink. Cette section explique avec un exemple plus avancé, la manière d'utiliser ces objets et de récupérer les données transmises.

Exemple d'utilisation de la balise <entry>

L'utilisation d'un objet entry est très simple, puisque les valeurs saisies peuvent être directement ajoutées à l'URL en tant que paramètres additionnels :

Code

```
<entry sizen="10 5"
   style="TextValueSmall"
   name="Saisie"
   default="Bonjour tout le monde !" />
<label posn="0 -20"
   style="CardButtonMedium"
   text="Envoyer"
   manialink="http://localhost/t1.php?valeur=Saisie" />
```

Script PHP de gestion de la saisie effectuée (t1.php) :

Code

```
<?php
// Récupération de la valeur saisie
$value = $_GET['valeur'];

// Escape the value to make our ManiaLink safe
$value = htmlspecialchars($value);

// Output the ManiaLink
header('Content-Type: text/xml');
echo <<<EOT
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
<timeout>0</timeout>
<label text="{$valeur}" />
</manialink>
EOT;
?>
```

Le ManiaLink contient une balise *<entry>* ayant le nom *Saisie*. Si ce nom apparaît dans une URL d'un lien, il est remplacé par la valeur actuelle de l'entrée. Donc, dans notre cas, le texte *Saisie* situé dans le lien du label est remplacée par l'information saisie par l'utilisateur. Ou, en d'autres termes : La saisie est passée à l'URL comme un paramètre nommé *value*.

Dans le script PHP, nous récupérons cette information via le tableau super-global $_GET puis générons un ManiaLink dans lequel cette valeur est retournée au sein d'un *label*.

Exemple d'utilisation de la balise <fileentry>

L'utilisation d'un objet fileentry est légèrement plus complexe car on ne peut passer le fichier téléversé comme paramètre de l'URL.

Code

```
<fileentry sizen="50 5" style="TextValueSmall"
   name="Map" folder="Maps"
   default="Sélectionner une piste..." />
<label posn="0 -20" style="CardButtonMedium"
   manialink="POST(http://localhost/t2.php?trk=Map,Map)"
   text="Envoyer la piste" />
```

Script PHP de gestion de la saisie effectuée (t2.php) :

Code

```
<?php

// Lecture du nom du fichier
$name = $_GET['trk'];
$file = file_get_contents('php://input');

// Sauvegarde du fichier transmis sur le serveur
// ATTENTION : Toujours verifier le fichier reçu
// On ne le fait pas ici, c'est un risqué de sécurité !
file_put_contents("maps/{$name}", $file);

// Lecture de la taille du fichier
$size = filesize("maps/{$name}");

// Echappement du nom de fichier
$name = htmlspecialchars($name);

// Envoi du ManiaLink
header('Content-Type: text/xml');
echo <<<EOT
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
<timeout>0</timeout>
<label text="Piste recue : {$name} (taille: {$size})" />
</manialink>
EOT;
?>
```

Le code XML est très similaire à celui de l'exemple précédent, mis à part le lien spécifié pour l'attribut *manialink* du label pour lequel nous pouvons lire *POST(http://localhost/t2.php?trk=Map,Map)*. Cette écriture signifie que nous envoyons le fichier choisi par l'utilisateur dans l'objet *fileentry* en utilisant la méthode POST à l'URL "http://localhost/t2.php" à qui on fournit en paramètre nommé *trk* le nom du fichier sélectionné.

Il faut lire la méthode POST ainsi : *POST(url, fichier)*. Ce qui est devant la virgule est l'URL appelé, ce qui est derrière est le fichier à transmettre. Et comme nous souhaitons récupérer le nom du fichier transmis, le plus simple est de le fournir en paramètre de l'URL.

Si on n'avait pas eu besoin du nom du fichier, on aurait pu écrire plus simplement : *POST(http://localhost/t2.php, Map)*.

En ce qui concerne le script PHP, il lit le contenu du paramètre *trk* afin de récupérer le nom du fichier, sauve ce dernier sur le disque du serveur puis renvoie un ManiaLink indiquant le nom du fichier reçu et sa taille.

Note

> Au risque de me répéter, n'oubliez surtout pas de vérifier le contenu du fichier transmis avant de le sauver sur disque, car l'URL est potentiellement publique, ce qui signifie que n'importe qui disposant de cet URL peut lui envoyer n'importe quel fichier, de n'importe quelle taille. Si vous ne voulez pas que votre serveur se transforme en repaire de Hackers, sécurisez-le. Vous pouvez par exemple vérifier que le fichier se termine par ".Map.Gbx" et que son contenu commence par les octets hexadécimaux "47 42 58 06 00 42 55 43 52" (le marqueur GBX..BUCR situé en tête des fichiers de pistes), voire d'utiliser l'une des nombreuses bibliothèques de ManiaPlanet pour valider le fichier.

Authentification avec ManiaConnect

Une bonne connaissance du PHP est requise.

ManiaConnect est un système avec lequel vous pouvez récupérer des informations détaillées sur le joueur qui visite actuellement ManiaLink – à condition qu'il ait accordé l'accès à ses données. Ces informations sont sécurisées et ne peuvent être manipulées du tout, donc, si vous récupérez par ce moyen le login "toto", vous pouvez être certain qu'il s'agit bien de ce joueur-là (ou qu'il s'agit de quelqu'un ayant piraté son compte).

ManiaConnect est l'une des composantes des Services Web ManiaPlanet (MPWS – ManiaPlanet Web Services). Les conditions préalables suivantes doivent être remplies par votre serveur Web pour pouvoir utiliser ces services et ManiaConnect en particulier :

- PHP 5.3 ou ultérieur,
- Extension CURL installée,
- Extension JSON installée.

Si votre serveur web supporte ces exigences, vous pouvez y allez et télécharger la dernière version du SDK MPWS pour PHP, puis :

Créer un utilisateur de l'API MPWS

Avant de pouvoir demander des données, nous devons créer un nouvel utilisateur de l'API avec lequel nous pourrons utiliser MPWS.

La création d'un nouvel utilisateur API est très simple :

1. Rendez-vous sur **http://player.maniaplanet.com** et connectez-vous avec votre compte ManiaPlanet.

2. Cliquez sur le menu *Advanced* dans la barre supérieure puis sur la commande de menu *Web Services*.

3. Cliquez alors sur *Create a new API user*, et poursuivez l'inscription jusqu'à vous retrouver sur la page de gestion de l'utilisateur d'API où se trouve le bouton *Create a ManiaConnect application*.

4. Cliquez sur le bouton *Create a ManiaConnect application* puis remplissez les champs du formulaire en prenant soin de bien remplir le champ *Redirection URI* dans lequel vous placerez l'URL de la page vers laquelle votre visiteur sera redirigé une fois qu'il aura donné son autorisation pour accéder à ses données.

Et c'est tout.

Demander des informations à ManiaConnect

Une fois l'utilisateur ManiaConnect et l'application ManiaConnect créés, nous pouvons écrire un ManiaLink utilisant MPWS pour afficher les informations du visiteur.

L'exemple ci-dessous montre comment afficher son pseudonyme :

Code

```
<?php
// Changer ces valeurs par celles de votre compte MPWS
define('MPWS_USER', 'mpws_user');
define('MPWS_PASS', 'mpws_pass');
define('MPWS_SCOPE', 'basic');

// Inclure les fichiers MPWS
require('autoload.php');

try {
```

```
  // Essayer de récupérer les infos du joueur
  $mpwsPlayer = new
    \Maniaplanet\WebServices\ManiaConnect\Player(
    MPWS_USER, MPWS_PASS);
  $player = $mpwsPlayer->getPlayer();

  if (!$player) {
    // Si $player vaut false, le joueur doit donner son
    // accord pour accéder à ses données, on récupère
    // son login
    $loginURL = htmlspecialchars(
      $mpwsPlayer->getLoginURL(MPWS_SCOPE));
    $label  = '<label style="CardButtonMedium"';
    $label .= 'text="Authentification" ';
    $label .= 'manialink="'.$loginURL.'" />';
  } else {
    // Si $player ne vaut pas false, il a donné son accord
    // pour accéder à ses données personnelles, on peut
    // donc récupérer son pseudonyme
    $nickname = htmlspecialchars($player->nickname);
    $label  = '<label sizen="100 5" halign="center" ';
    $label .= 'text="Your Nickname: '.$nickname.'" />';
  }
} catch (\Maniaplanet\WebServices\Exception $e) {
  // L'accès à ManiaConnect ou aux WebServices a échoué
  // On affiche juste un message d'erreur
  $msg  = "[{$e->getHTTPStatusCode()} ";
  $msg .= "{$e->getHTTPStatusMessage()}] ";
  $msg .= "{$e->getMessage()}";
  $msg = htmlspecialchars($msg, ENT_QUOTES, 'UTF-8');
  $label  = '<label sizen="100 5" halign="center" ';
  $label .= 'text="Exception: '.$message.'" />';
}
// Envoi du ManiaLink
header('Content-Type: text/xml; charset=utf-8');
echo <<<EOT
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
<timeout>0</timeout>
{$label}
</manialink>
EOT;
?>
```

La classe *Player* doit être instanciée avant la première émission d'informations à destination de l'utilisateur, car elle initialise une session (en utilisant les mécanismes de session internes de PHP).

L'accès aux informations du joueur connecté s'effectue avec un appel à la méthode *getPlayer()* qui retourne les informations personnelles du joueur s'il a donné son autorisation, ou qui retourne *false* dans le cas contraire. Avec un test sur la valeur retournée, nous sommes en mesure de décider si le pseudonyme peut être affiché, ou si nous devons d'abord obtenir son consentement.

Pour cela, nous devons appeler *getLoginURL()* pour obtenir l'URL du ManiaLink permettant au joueur de donner ou de refuser de donner son accord pour que nous puissions accéder à ses informations. Nous fournissons cet URL en tant que bouton que l'utilisateur pourra cliquer et c'est terminé : une fois que le joueur aura donné (ou pas) son accord, il sera automatiquement redirigé sur notre page PHP.

La portée de la demande d'informations doit être spécifiée lors de l'appel de la méthode *getLoginURL()*. Cette portée permet simplement de spécifier le type d'informations dont nous avons besoin pour notre application. Elle peut être l'une des suivantes :

- **Basic** : Permet de récupérer des informations de base, tels que le login, le pseudonyme et la zone avec la méthode *getPlayer()*.
- **buddies** : Permet de récupérer les amis avec la méthode *getBuddies()*.
- **dedicated** : Permet de récupérer la liste des serveurs dédiés créés avec le compte du joueur avec la méthode *getDedicated()*.
- **email** : Permet de récupérer l'adresse e-mail du joueur avec la méthode *getEmail()*.
- **manialinks** : Permet de récupérer la liste des ManiaLinks enregistrés avec la méthode *getManiaLinks()*.
- **online_status** : Permet de récupérer le statut de connexion avec la méthode *getOnlineStatus()*.

Pour spécifier plusieurs portée dans l'appel à *getLoginURL()*, il suffit de les séparer par des espaces.

Note

> Pour éviter un refus peut-être justifié, il est conseillé de ne demander que la ou les portées qui vous sont absolument nécessaires et de prévoir une alternative en cas de refus.

Révoquer l'accès à ManiaConnect

Si vous souhaitez révoquer l'accès à ManiaConnect pour un, plusieurs ou la totalité de vos ManiaLinks, il vous suffit de retourner sur votre page joueur où vous pourrez gérer votre compte utilisateur de l'API.

Postface
À vous de jouer !

Nous voici arrivés, après cinq chapitres, à l'issue de de didacticiel concernant les ManiaLinks de ManiaPlanet. C'est maintenant à vous de jouer, de vous imprégner de ce concentré de connaissances afin de l'appliquer à vos propres ManiaLinks. Vous pouvez le contacter en vue de me faire part de vos commentaires, de vos remerciements, de vos suggestions et pour me remonter les éventuelles erreurs que vous aurez découvertes dans ce manuel.

Pour terminer, je tiens à remercier tous ceux qui m'ont soutenu dans la rédaction de ce didacticiel, Nadeo bien sûr, mais aussi tous ceux qui m'ont apporté leur aide dans les forums ou ailleurs. Et un remerciement spécial à *FT»kastun*, *seeba* et *destroflyer*, qui a lu de manière critique le didacticiel et a souligné toutes les erreurs, en plus de m'aider à y insérer les derniers détails. Sans leur soutien, ce didacticiel n'aurait jamais été aussi complet.

FT»Marcel

Le 13 février 2012

http://forum.funtrackers.net