

# Introduction API C++ Maya

**lionel.reveret@inria.fr**

2010-11

## Comparaison entre script MEL et API C++

Pour tout ce qui est couvert en commun entre le script MEL et l'API C++ (création d'objet, gestion de l'animation), l'API est très sensiblement plus rapide à l'exécution. L'API permet en outre d'accéder à des fonctionnalités que le script MEL ne permet pas : création de nouveaux outils liés à la souris, import/export de nouveaux formats de fichier, calcul de shaders, etc. L'API peut fournir des parties de visualisation en OpenGL. Elle donne même accès au contexte OpenGL de la visualisation interactive. Le MEL reste cependant l'unique moyen de construire un Interface Graphique.

Comme le script MEL, l'API permet la création de nouveaux node du DG et de nouvelles commandes.

L'utilisation de l'API se décline de trois manières : des plug-ins intégrables sous l'interface générale de Maya (chargement de bibliothèque dynamique), d'application de type client/serveur pour la capture de mouvement (on développe un exécutable qui sera serveur, auquel Maya se connecte en tant que client) et d'application indépendante de l'application de Maya, se contentant d'utiliser des fonctionnalités de Maya via le chargement d'une bibliothèque dynamique spécifique à Maya.

La documentation développeur présente un document utilisateur (API Guide) et une liste exhaustive des classes de l'API (API Classes).

Ce document liste quelques exemples distribués avec l'installation régulière de Maya.

## Préparatifs à la compilation

Cette procédure s'applique à tous les plug-ins à charger.

1. Spécifier le chemin des plug-ins et des scripts via le fichier Maya.env à mettre dans le repertoire maya de l'utilisateur

```
MAYA_SCRIPT_PATH = Z:\maya\devkit\scripts;Z:\maya\devkit\plug-ins
```

```
MAYA_PLUG_IN_PATH = Z:\maya\devkit\plug-ins
```

```
XBMLANGPATH = Z:\maya\devkit\plug-ins\%B
```

2. Mettre à jour la variable MAYA\_LOCATION dans ses variables d'environnement windows, pour pouvoir la référencer ensuite dans les Makefile

*Typiquement, MAYA\_LOCATION=C:\Program Files\Alias\Maya7.0*

3. Copier les exemples indiqués ci-dessous depuis C:\Program Files\Alias\Maya7.0\devkit\plug-ins, applications, et mocap

4. Générer les fichiers .mll (en fait une dll) dans Z:\maya\devkit\plug-ins

5. Via les propriétés des projets, modifier les include et les lib pour inclure

```
$(MAYA_LOCATION)/include et $(MAYA_LOCATION)/lib
```

6. Dans Command Line du Linker, ajouter /export:initializePlugin /export:uninitializePlugin

7. Attention, seulement pour le plug-in MayaClockServer, changer winsock2 en winsock

8. On vérifie le chargement d'un plug-in sous Maya via le plug-ins manager :

Windows>Settings/Preferences>Plug-ins manager

Tout ceci concerne la compilation sous Windows. Cependant, l'API est entièrement portable sous Linux, Mac et SGI: le meme code peut se recompiler. Pour les détails sous Linux, les fichiers projets VisualC++ sont des Makefile, les libraries dynamiques mll sont des fichiers de type .so .

## Exemples de Plug-ins

### Commandes

#### helloWorldCmd

Juste pour commencer. La fonction DeclareSimpleCommande masque beaucoup de choses. En changeant le texte affiché, cet exemple permet de vérifier la compilation.

*Compiler et essayer la commande helloWorld*

*Retrouver ce que "cache" la macro DeclareSimpleCommande*

#### scanDagCmd

Montre une dérivation complète de la classe MPxCommand afin d'ajouter de nouvelles fonctions.

On voit les points d'entrées du plug-in, et quelques fonctions clé d'une commande (doIt)

A noter, comme en MEL, il existe des fonctions de traitements des chaînes de caractères.

*commande scanDag* (la sortie n'est visible que dans l'output window à cause des cout)

*A titre d'exercice, modifier le programme pour lister aussi les joints.*

#### helixCmd

Montre la création de formes NURBS

La fonction doIt est toujours présente mais sa déclaration est implicite dans DeclareSimpleCommande

*commande "helix [ -r #] [ -p #]"*

*analyser le code de l'interface graphique helixCmd.mel : comment se fait le lien entre le script et le plug-in*

?

#### spiralAnimCurveCmd

Montre l'accès aux objets via les classes MFn.

Le but d'un tel accès est de ne pas avoir de risque de modification de la scène avec un accès direct à MObject.

Cet exemple montre la manipulation des courbes d'animations.

*Sélectionner un objet, puis exécuter la commande spiralAnimCurve*

#### lepTranslator

Un exemple de définition d'import/export de fichier, avec les fonctions type : reader et writer.

Un fichier d'options accessibles est donné via un mel (lepTranslatorOpts.mel) : il renvoie le string de optionString dans reader et writer.

*Exporter et importer une scène quelconque.*

#### moveTool

Premier exemple de nouvel outil lié à la souris :

dès qu'un objet est sélectionné à la souris, l'outil permet d'accéder directement à une translation (tant que l'outil reste l'outil courant, notion de contexte d'outil) dans les vues orthographiques.

MPxContextCommand moveContextCmd, crée le contexte d'utilisation de la souris

MPx(Selection)Context moveContext, gère les événements et les envoie à l'outil

MPxToolCommand moveCommand, exécute l'outil

*Le script mel moveTool.mel installe l'appel à l'outil dans une shelf Shelf1 (à créer avec le shelf editor, ou modifier le script pour s'insérer dans la shelf Custom)*

*Appliquer alors le nouvel outil à un objet sélectionné.*

**moveManip**

Meme chose qu'au-dessus, avec en plus un manipulateur Maya, se comportant comme un objet géométrique inscrit dans la scène.

MPxManipContainer moveManip.

Il existe différents type de manipulateur pré-établi (voir doc), ici une distance est liée à la taille de l'objet manipulé.

*Executer via le script mel moveToolManip.mel d'installation dans une shelf shelf1 (à créer avec le shelf editor, ou modifier le script pour s'insérer dans la shelf Custom)*

**helixTool**

Un outil un peu plus complexe, qui aboutit au dessin de l'ellipse. Le dessin de la cage de l'outil se fait en OpenGL.

MPxContext helixContext

MPxContextCommand helixContextCmd

MPxToolCommand helixTool

*script mel helixTool.mel d'installation dans une shelf shelf1 (à créer)*

**quadricShape**

Exemple de création d'une nouvelle forme géométrique intégrable sous Maya. Celle-ci s'appuie sur les gluQuadric de OpenGL

MPxSurfaceShape impose certaines fonctions liées à la géométrie.

MPxSurfaceShapeUI impose les interfaces de dessin.

*Identifier où se déroule le rendu OpenGL et modifier le (changer la couleur par exemple)*

**circleNode**

Un premier node tout simple qui fournit le cosinus et le sinus de son entrée

Il faut définir les entrées et les sorties. Elles sont liées par la fonction compute.

Le système d'évaluation du DG déclenche les évaluations par propagations de flags.

*Le script circleNode.mel crée un menu et anime une sphère tournante grace au node.*

**animCubeNode**

Ce noeud illustre la création de mesh. Le mesh entier est une sortie.

Une sortie peut etre un objet complexe, pas seulement un attribut

*script animCubeNode.mel de création d'animation (connecte une shape à la sortie du noeud, et le temps à l'entrée)*

**interpShader**

Un exemple de noeud de shading, crée un halo autour de l'objet par rapport au point de vue.

A tester sur un objet de type nurbs

*Créer un objet NURBS avec un material phong*

*sphere -n sphere1 -r 1;*

*createNode interShader;*

*connectAttr interShader1.outColor phong1.color;*

**simpleEmitter****simpleSpring**

Ajoute des fonctionnalités aux noeuds MPxEmitterNode et MPxSpringNode

*scripts d'essai simpleEmitter.mel et simpleSpring.mel*

**MayaClockDemo**

Exemple de connexion externe au logiciel.

*Compiler et lancer le serveur externServer.c (situé dans C:\Program Files\Alias\Maya7.0\devkit\mocap).*

*Executer le script MayaClockDemo.mel (situé dans C:\Program Files\Alias\Maya7.0\scripts\others).*

*Constater les connexions avec le Device Editor et les divers options de filtrage et enregistrement.*

### **Maya en externe**

Montre l'utilisation de la lib pour traiter des fichiers Maya et créer des formes, sans avoir à ouvrir l'interface.

*asciiToBinary* convertit un fichier du format texte au format binaire de Maya.

*surfaceCreate* crée une surface NURBS et génère un fichier de scène Maya.