Initiation à la programmation COFFEE. (partie 4.)

Par Tengaal (tengaal@libertysurf.fr) 01/07/2002.

Les containers

Les « containers » représentent tout simplement les paramètres d'un objet. Par exemple, si vous créez un cube primitif dans Cinéma 4D, il va être ajouté dans le gestionnaire d'objet avec son nom et une icône. En ouvrant cette icône, on accède aux paramètres du cube, et on peut y définir le nombre de segments sur les axes X, Y et Z, ainsi que l'activation ou non d'un biseau pour arrondir les arêtes (avec en paramètres le rayon du biseau et le nombre de segments) et enfin la possibilité de séparer les surfaces.

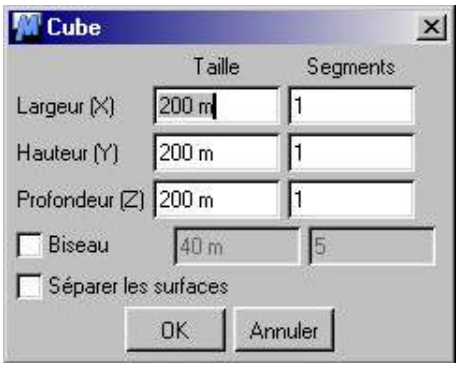
Cet ensemble de paramètres sont accessibles par le COFFEE, car ils sont contenus dans le container de l'objet « Cube » et pour y accéder il suffit donc de procéder ainsi :

```
var mon_container = Mon_cube -> GetContainer() ;
```

Bien sûr « Mon_cube » représente l'objet cube. (voir Partie 3 :accéder et pointer sur un objet.)

Le principe

Pour comprendre comment utiliser les containers, nous allons étudier le cas d'un cube primitif, dont les paramètres sont accessibles dans l'éditeur de Cinema4D en affichant la fenêtre suivante :



Le container de notre cube est donc l'ensemble de ces paramètres, et nous pouvons accéder à chacun de ces paramètres grâce à des valeurs définies par Cinema4D, qui permettent de lire ou de modifier chacune d'elles. Voici la liste des valeurs de Cinema4D qui donnent accès à chaque paramètre du cube :

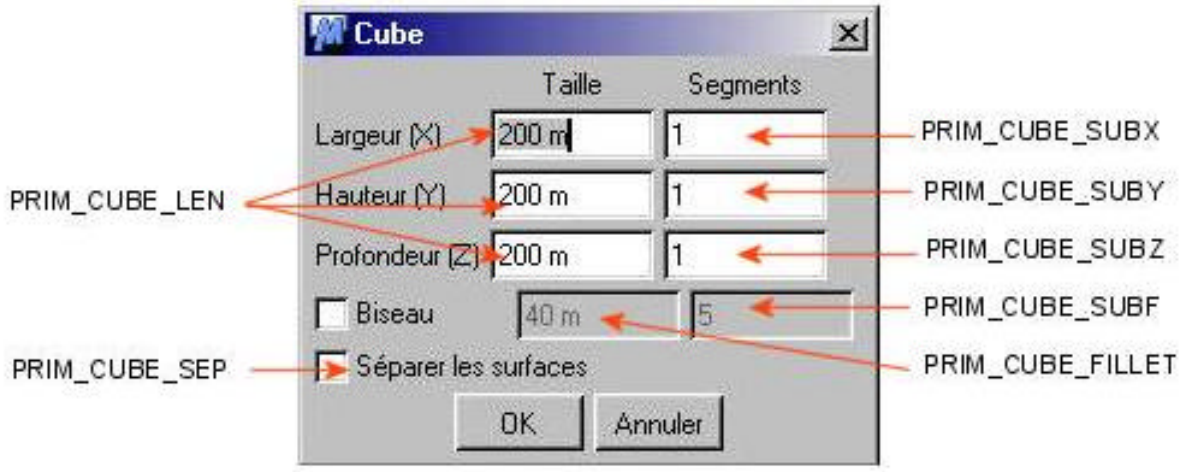
PRIM_CUBE_LEN	représente un vecteur (x,y,z)	vector
PRIM_CUBE_SUBX	représente une valeur entière	int
PRIM_CUBE_SUBY	représente une valeur entière	int
PRIM_CUBE_SUBZ	représente une valeur entière	int
PRIM_CUBE_FILLET	représente une valeur à virgule	float
PRIM_CUBE_SUBF	représente une valeur entière	int
PRIM_CUBE_SEP	représente une valeur booléenne	bool

Ces valeurs sous forme de noms (identifiant ou ID), donnent accès aux paramètres d'un cube primitif, chaque objet de Cinema4D (objet polygonal, objets Nurbs, déformants…) a ses propres paramètres donc ses propres valeurs d'accès. Dans la documentation du SDK COFFEE fournie par Maxon sur le site http://www.plugincafe.com , toutes ces valeurs sont données et expliquées pour chaque objet.

Nous allons voir maintenant à quoi correspondent ces valeurs par rapport aux paramètres de notre cube, puis comment y accéder par le COFFEE.

En pratique …

Voici le schéma qui donne la correspondance entre les paramètres d'une fenêtre et leur identifiant (ID) en COFFEE.



Les choses commencent à s'éclaircir petit à petit ! Voyons tout d'abord comment accéder au container d'un objet.

Restons donc dans le cas de notre cube, imaginons que notre programme COFFEE pointe sur un cube de type primitif de notre scène 3D et que nous souhaitons lui donner une largeur de 100m, une hauteur de 50m et une profondeur de 300m. Enfin nous voulons que notre cube soit subdivisé en 5 segments pour chaque face.

```
main(doc,op)
{
// On pointe sur notre cube qui s'appelle "Cube"
var mon_cube = doc -> FindObject("Cube") ;
//si "Cube" n'est pas trouvé, mon_cube=NULL donc on quitte le programme
if ( !mon_cube) return ;

//On accède au container du cube avec la fonction Objet -> GetContainer().
var mon_container = mon_cube -> GetContainer() ;
//si il y a un problème et que le container n'est pas accessible, on quitte.
if ( !mon_container) return ;

//Nous allons fixer ces nouvelles valeurs dans le container avec la fonction
// Container -> SetData(ID_du_paramètre, Nouvelle_valeur)
mon_container -> SetData(PRIM_CUBE_LEN,vector(100,50,300)) ;

//Ensuite on passe à 5 la subdivision en segments
mon_container -> SetData(PRIM_CUBE_SUBX,5) ;
mon_container -> SetData(PRIM_CUBE_SUBY,5) ;
mon_container -> SetData(PRIM_CUBE_SUBZ,5) ;

//Maintenant que les paramètres sont modifiés, il faut redonner à notre cube
//ce nouveau container avec la fonction Objet -> SetContainer (Container).
Mon_cube -> SetContainer(mon_container) ;

}  //Et voilà !
```

La fonction **SetData(ID_du_paramètre, Nouvelle_valeur)** permet donc de modifier chaque paramètre, la fonction **GetData(ID_du_paramètre)** permet de lire le paramètre souhaité, elle renvoie sa valeur :
Ex : var separation = mon_container -> GetData(PRIM_CUB_SEP) ;
PRIM_CUB_SEP correspond à l'activation de la séparation de surface, c'est une valeur de type booléen donc si separation=TRUE cela signifie que l'option est activée sinon si separation=FALSE, cela signifie que l'option est désactivée.

Liste des ID des paramètres d'objets primitifs

Objet Primitif	Identifiant des paramètres	Explication
Valeurs communes	PRIM_SLICE [bool]	Portion
(cylindre, cône, disque,	PRIM_SLICEFROM [float]	Angle de début
tube, capsule,	PRIM_SLICETO [float]	Angle de fin(>=angle de début)
réservoir…)	PRIM_REGULAR [bool]	Grille régulière
	PRIM_REGULARWIDTH [float]	Grille régulière en largeur
	PRIM_AXIS [int]	Direction (0=+X, 1=-X, 2=+Y, 3=-Y, 4=+Z, 5=-Z)
PRIMITIVE_CUBE	PRIM_CUBE_LEN [vector]	Dimensions (>=0.0)
	PRIM_CUBE_SEP [bool]	Surfaces séparées(sans biseau)
	PRIM_CUBE_SUBX [int]	X subdivision (>=0.0)
	PRIM_CUBE_SUBY [int]	Y subdivision (>=0.0)
	PRIM_CUBE_SUBZ [int]	Z subdivision (>=0.0)
	PRIM_CUBE_FILLET [float]	Rayon biseau (0.0<=rayon<=longueur/2)
	PRIM_CUBE_SUBF [int]	Subdivision biseau(0=sans biseau)
PRIMITIVE_SPHERE	PRIM_SPHERE_RAD [float]	Rayon (>=0.0)
	PRIM_SPHERE_SUB [int]	Subdivision (>0)
	PRIM_SPHERE_TYPE [int]	Type (0=Standard, 1=Tetra, 2=Hexa, 3=Octa, 4=Icosa)
	PRIM_SPHERE_PERFECT [bool]	Sphère parfaite
PRIMITIVE_PLATONIC	PRIM_PLATONIC_RAD [float]	Rayon extérieur (>=0.0)
	PRIM_PLATONIC_SUB [int]	Subdivision (>0)
	PRIM_PLATONIC_TYPE [int]	Type (0=Tetragone, 1=Hexagone, 2=Octagone, 3=Dodeca, 4=Icosa, 5=Bucky)
PRIMITIVE_CONE	PRIM_CONE_TRAD [float]	Rayon du haut (>=0.0)
	PRIM_CONE_BRAD [float]	Rayon du bas (>=0.0)
	PRIM_CONE_HEIGHT [float]	Hauteur (>=0.0)
	PRIM_CONE_HSUB [int]	Subdivision en hauteur (>0)
	PRIM_CONE_CSUB [int]	Segments de couvercles(>0)
	PRIM_CONE_SEG [int]	Subdivision circulaire (>2)
	PRIM_CONE_TYPE [int]	Type (0=Sans couvercles, 1=avec, 2=arrondi en haut, 3=arrondi en bas, 4=arrondi en haut et en bas)
	PRIM_CONE_TFILLET [float]	Biseau haut (>=0.0)
	PRIM_CONE_BFILLET [float]	Biseau bas
	PRIM_CONE_TFILLETY [float]	Biseau haut Y (>=0.0)
	PRIM_CONE_BFILLETY [float]	Biseau bas Y (>=0.0)
	PRIM_CONE_FSUB [int]	Subdivision biseau(>0)
	PRIM_CONE_TFILANGLE [float]	Angle biseau haut
	PRIM_CONE_BFILANGLE [float]	Angle biseau bas
PRIMITIVE_TORUS	PRIM_TORUS_OUTERRAD [float]	Rayon extérieur (>=0.0)
	PRIM_TORUS_INNERRAD [float]	Rayon intérieur (>=0.0)
	PRIM_TORUS_CSUB [int]	Subdivision de section (>2)
	PRIM_TORUS_SEG [int]	Subdivision circulaire (>2)
PRIMITIVE_DISC	PRIM_DISC_IRAD [float]	Rayon intérieur (>=0.0)
	PRIM_DISC_ORAD [float]	Rayon extérieur (>=Inner radius)
	PRIM_DISC_CSUB [int]	Subdivision couvercle(>0)
	PRIM_DISC_SEG [int]	Subdivision circulaire (>2)
PRIMITIVE_TUBE	PRIM_TUBE_ORAD [float]	Rayon extérieur (>=0.0)
	PRIM_TUBE_IRAD [float]	Rayon intérieur (>=0.0)
	PRIM_TUBE_HEIGHT [float]	Hauteur(>=0.0)
	PRIM_TUBE_HSUB [int]	Subdivision en hauteur(>0)
	PRIM_TUBE_CSUB [int]	Subdivision couvercle(>0)

	PRIM_TUBE_SEG [int]	Subdivision circulaire (>0)
PRIMITIVE_FIGURE	PRIM_FIGURE_HEIGHT [float]	Hauteur
	PRIM_FIGURE_SUB [int]	Subdivision (>2)
PRIMITIVE_PYRAMID	PRIM_PYRAMID_LEN [vector]	Dimensions
	PRIM_PYRAMID_SUB [int]	Segments (>0)
PRIMITIVE_PLANE	PRIM_PLANE_WIDTH [float]	Largeur (>=0.0)
	PRIM_PLANE_HEIGHT [float]	Hauteur (>=0.0)
	PRIM_PLANE_SUBW [int]	Subdivision en largeur (>0)
	PRIM_PLANE_SUBH [int]	Subdivision en hauteur (>0)
PRIMITIVE_FRACTAL	PRIM_FRACTAL_LEN [float]	Dimensions
	PRIM_FRACTAL_SUBW [int]	Subdivision en largeur (>0)
	PRIM_FRACTAL_SUBH [int]	Subdivision en hauteur (>0)
	PRIM_FRACTAL_ROUGH [float]	Sillons grossiers(0.0<=x<=1.0)
	PRIM_FRACTAL_FINE [float]	Sillons fins (0.0<=x<=1.0)
	PRIM_FRACTAL_SCALE [float]	Echelle (0.0<=x<=1.0)
	PRIM_FRACTAL_BLEVEL [float]	Niveau de la mer (0.0<=x<=1.0)
	PRIM_FRACTAL_TLEVEL [float]	Niveau du plateau (0.0<=x<=1.0)
	PRIM_FRACTAL_MULTIFRACTAL[bool]	Fonction multifractale
	PRIM_FRACTAL_BORDERS [bool]	Bordures au niveau de la mer
	PRIM_FRACTAL_SPHERICAL [bool]	Forme sphérique
PRIMITIVE_RELIEF	PRIM_RELIEF_LEN [float]	Dimensions (>=0.0)
	PRIM_RELIEF_SUBW [int]	Subdivision en largeur (>0)
	PRIM_RELIEF_SUBH [int]	Subdivision en hauteur (>0)
	PRIM_RELIEF_TEXTURE [string]	Nom de la texture
	PRIM_RELIEF_SPHERICAL [bool]	Forme sphérique
	PRIM_RELIEF_BLEVEL [float]	Niveau de la mer (0.0<=x<=1.0)
	PRIM_RELIEF_TLEVEL [float]	Niveau du plateau (0.0<=x<=1.0)
PRIMITIVE_POLY	PRIM_POLY_WIDTH [float]	Largeur (>=0.0)
	PRIM_POLY_HEIGHT [float]	Hauteur (>=Largeur)
	PRIM_POLY_SUB [int]	Subdivision (>0)
	PRIM_POLY_TRIANG [bool]	Type (TRUE=Triangle, FALSE=Carré)

PRIMITIVE_CYLINDER (Basé sur PRIMITIVE_CONE)

PRIMITIVE_CAPSULE (Basé sur PRIMITIVE_CONE)

PRIMITIVE_OILTANK (Basé sur PRIMITIVE_CONE)

Je vous encourage donc à consulter la documentation du SDK COFFEE de Maxon pour connaître les identifiants de tous les autres objets (matériaux, pistes animations, lumières, os et objets déformants, caméras …)

La possibilité de pouvoir modifier les paramètres d'objets en COFFEE permet ainsi d'obtenir les effets les plus variés comme par exemple l'utilisation de la position d'un objet pour définir l'intensité lumineuse d'un spot ciblé sur lui en fonction de la distance qui les sépare, ou bien encore l'ajustement automatique des dimensions d'un objet en fonction de sa position sur l'axe Y….

Comprendre l'utilisation des containers permet déjà les programmations les plus spectaculaires sans aucune limite, et offre la possibilité de créer ses propres outils adaptés aux contraintes techniques d'un projet 3D.

Vous avez maintenant en main toutes les connaissances de base en COFFEE !
Dans les prochaines parties, nous verrons le cas des matériaux, des pistes et des clés d'animation ainsi que le principe du temps dans Cinema4D, enfin, quelques exemples pratiques de scripts COFFEE qui mettent en œuvre ces bases de programmation pour des situations précises.