Initiation à la programmation COFFEE. (partie 2.)

Par Tengaal (tengaal@libertysurf.fr) 19/03/2002.

Présentation

Cette seconde partie va nous faire découvrir le monde merveilleux des boucles avec les instructions **for**, **do-while**, et **break**. Puis nous ferons un récapitulatifs des opérateurs logiques utilisables avec les tests de conditions et les opérateurs arythmétiques, enfin nous terminerons avec les différents types de variables (nombres, texte, tableaux, objets...)

Les boucles

La boucle for permet de répéter un groupe d'instructions un nombre X de fois.

Utilisation :

```
for (valeurInitiale ;condition ;incrémentation)
{
  instructions ;
}
```

Explication : valeurInitiale représente la valeur de départ du comptage.
Condition doit être vraie pour exécuter le groupe d'instructions.
Incrémentation définit comment le compteur doit être augmenté ou diminué.

Exemple :

```
var i ;                    // création de la variable de comptage
for (i=0 ;i<50 ;i++)   // compteur de 0 à 49 par 1 unité soit 50 répétitions
{
   //instructions à exécuter
}
```

Explication : le compteur i commence à la valeur 0.
i doit être inférieur à 50 pour que la boucle se fasse.
i++ équivaut à i=i+1, à chaque passage i est augmenté de 1 unité.

Donc i va prendre la valeur 0 à 49, et les instructions seront exécutées 50 fois.

La boucle **do** répète un groupe d'instructions en permanence jusqu'à ce qu'un événement de cette boucle provoque la sortie soit avec **while(condition)** associé à do ou bien si la commande **break** est exécutée dans la boucle.

Utilisation :

```
do
{
  if (condition) instructions ;
  else break ;
}
```

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

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

Explication : Ces 3 possibilités ont le même effet, c'est à dire que les boucles ne sont pas comptées, la répétition se fait si la condition est vraie, dès que la condition est fausse, on sort de la boucle et on passe à la suite du programme.

Exemple :

```
var a=10 ;
var b=1 ;
while (b<101)   // on sort de la boucle si b est supérieur ou égal à 101
{
  b=b+a ;        // s'écrit également b+=a
}
```

Explication : A chaque boucle b est recalculé ce qui ici signifie que b est augmenté de 10
Le premier passage se fait pour b=1 et le calcul de la nouvelle valeur de b sera testé au passage suivant, dans notre cas la boucle va être exécutée 10 fois car b va prendre les valeurs 1, 11, 21, 31 …. 91. A 101, la condition est fausse.

Les opérateurs

Les opérateurs sont des caractères qui définissent le liens entre deux valeurs, on distingue parmi eux les opérateurs logiques qui permettent de savoir si le liens entre les deux valeurs est vrai ou faux (TRUE ou FALSE) et les opérateurs arythmétiques qui servent au calculs entre les valeurs.

Les opérateurs logiques : **==** vérifie l'égalité entre 2 valeurs.
!= vérifie l'inégalité entre 2 valeurs.
< vérifie si une valeur est strictement supérieure à une autre.
<= vérifie si une valeur est inférieure ou égale à une autre.

Exemple :
```
if (a==b) instruction ;   // a doit être égal à b
if (a !=b) instruction ;  // a et b doivent être différents
if (a<b) instruction ;    // a doit être inférieur à b
if (a<=b) instruction ;   // a doit être inférieur ou égal à b
```

NB : 3 autres opérateurs permettent de tester les conditions entres elles essentiellement pour les instructions if, une fois appliquées, elles donnent une condition globales.

&& est un ET logique, les 2 conditions doivent être vraies.
|| est un OU logique, une des 2 conditions doit être vraie.(2 caractères Alt + '6')
! est un NON logique qui s'applique à une seule condition qui doit être fausse.

Exemples :
```
if ((condition1) && (condition2)) instructions ; // les 2 conditions doivent être vraies
If ((condition1) || (condition2)) instructions ;  // une des 2 conditions doit être vraie
if (!(condition)) instructions ;                  // la condition doit être fausse
```

Les opérateurs arythmétiques : **=** pour attribuer une valeur à une variable
+, -, /, * addition, soustraction, division et multiplication
+= la variable vaut une valeur + elle-même.
-= la variable vaut une valeur – elle-même
/= la variable vaut une valeur / par elle-même
***=** la variable vaut une valeur * elle-même

Exemple :
```
a+=b équivaut à a=b+a
a-=b  équivaut à a=b-a
a/=b  équivaut à a=b/a
a*=b  équivaut à a=b*a
```

Les types de variables

On peut distinguer 4 catégories de variables : les nombres, les textes, les tableaux et les objets de Cinema 4D.
En général, on n'a pas besoin de définir le type d'une variable car lorsqu'on la créé avec la fonction var MaVariable; son type sera définit dès qu'on lui affectera une valeur. De la même manière, si on initialise la variable au moment de sa création, son type est définit tout de suite (valeur numérique, chaîne de caractères, …)

Exemple :

```
var MaVariable ;

main(doc,op)
{
  MaVariable=op ->GetPosition();  //on prend la position (x,y,z) de l'objet op
}
```

Explication : MaVariable est créée sans être définie, puis dans la fonction main on lui attribut la position de l'objet op (objet auquel est associée l'expression COFFEE), cette position dans l'espace est en fait un type vector qui stock les 3 valeurs x,y et z.
MaVariable est donc automatiquement définie comme un vecteur. Dans ce cas précis les coordonnées sont MaVariable.x, MaVariable.y et Ma Variable.z. Nous verrons en détails l'utilisation des vecteurs un peu plus tard…

Pour initialiser une variable numérique ou texte il suffit donc de lui attribuer le nombre ou le texte :

```
var MaValeur=12345;                 // variable de type integer
var MonTexte= "blablabla";          // variable de type string
```

Pour initialiser les tableaux ou les objets Cinema 4D il faut utiliser la commande new :

```
var MonTableau=new(array,10) ;     // création d'un tableau de 10 éléments
var MonObjet=new(SplineObjet) ;    // création d'un objet Spline
```

Accès aux éléments d'un tableau : Lire le 1e élément MonElement=MonTableau[0] ;
Lire le 6e élément MonElement=MonTableau[5] ;

On peut bien sûr créer des tableaux à plusieurs dimensions:

```
var MonTableau=new(array,10,2) ;   // le tableaux contient 10*2 éléments
var MaValeur=MonTableau[9,1] ;
```

Il est impossible de changer le type d'une variable, une fois qu'elle est initialisée, son type lui est attribué définitivement, c'est à dire que si vous créez et initialisez une variable avec une valeur numérique vous ne pourrez pas ensuite lui attribuer un texte ou un objet C4D.

Fin de cette seconde partie.