Séquence 4 : Les boucles
for do while

I. Généralités

Les boucles sont des techniques qui permettent de répéter les mêmes instructions plusieurs fois.
Tout comme pour les conditions, il y a plusieurs façons de réaliser des boucles.
Dans tous les cas, le schéma est toujours le même :
On répète n fois les mêmes instructions.

II.La boucle *while*

II.1.Structure

La boucle while s'écrit de la manière suivante :

```
while (condition)
        {
                instruction 1;
                instruction 2;
                instruction 3;
                ...;
                instruction n;
        }
```

Le terme *while* se traduit par tant que. La structure répétitive s'exécute selon les principes suivants :

- Tant que l'expression à l'intérieur des parenthèses reste vraie, la ou les instructions composant la boucle sont exécutées.
- Le programme sort de la boucle dès que l'expression à l'intérieur des parenthèses devient fausse.

II.2.Application

Vous allez réaliser un application qui permet de créer et positionner aléatoirement 10 billes sur la scène.
Pour cela, vous allez créer un nouveau projet AS3 appeler repeterObjet.
Ensuite recopier le code ci-dessous pour placer les 10 billes sur la scène.

```
public class Main extends Sprite
        {
                /*****************************************
                 * Déclaration des variables de la classe
                 *****************************************/
                //Declare un objet graphique qui charge une image
                private var monSprite:ChargerSprite = new ChargerSprite("images/ballon.png");

                //Declare un objet pour gerer un son
                private var monSon:GererSon = new GererSon("sons/applaudissement.mp3");


                //Variables pour gérer la largeur et la hauteur de la scène
                private var largeurEcran : int ;
                private var hauteurEcran : int ;

                //Crée un objet zone de texte
```

```
private var zoneText : ZoneText = new ZoneText(30, 200);

//Crée un objet zone de texte
private var inputText : InputText = new InputText(30, 400);

//Crée un objet bouton
private var myButton : ButtonColor = new ButtonColor("Valider", 100 , 0xff0000, 0xffffff, 20);

//déclaration de l''objet graphique bille
private var bille : Sprite ;

public function Main()
{
	//récupérer la largeur et la hauteur de la scène
	largeurEcran = stage.stageWidth ;
	hauteurEcran = stage.stageHeight ;


	//Création de l'écouteur d'évènement sur les touches
	stage.addEventListener(KeyboardEvent.KEY_DOWN, gestionDesTouches );

	//met l'objet inputText sur la scène
	addChild(inputText);
	//le jeu de caractères que l'utilisateur peut rentrer dans le champ inputText
	inputText.restrict("0-9");

	//met l'objet myButton sur la scène
	addChild(myButton);
	//le jeu de caractères que l'utilisateur peut rentrer dans le champ inputText
	myButton.x = inputText.width;
	myButton.buttonMode = true;
	myButton.addEventListener(MouseEvent.CLICK, cliqueBouton);

	//met l'objet zone de texte sur la scène
	addChild(zoneText);
	zoneText.y = inputText.height;


	//ajoute un texte dans la zone de texte
	zoneText.addText("Hello World ;-)");

	/*****************************************/
	// Ecrire la structure while en dessous :
	/*****************************************/
	///dessin des 10 billes

	// déclarer et initialiser les variables
	var i : int = 0;
	var nombreDeBille : int = 10;

	while (i < nombreDeBille) //tant que i < nombreDeBille tratite les
	{	trace("while",i);			//instructions entre les accolades
		i = i + 1 ;
		//créer un nouvel objet bille
```

```
                bille = new Sprite();
                bille.graphics.beginFill(0xFF0000);
                bille.graphics.drawCircle(0, 0, 15);
                addChild(bille);
                //tester

                //place au hasard la bille sur la scène
                bille.x = largeurEcran * Math.random();
                bille.y = hauteurEcran * Math.random();
                //tester

                //associe à chacune des billes un évènement clique
                bille.addEventListener(MouseEvent.CLICK, billeCliquer);

        }//fin de la boucle while
        // tester ! Ctrl Enter


}//fin de la méthode Main

/**********************************************
 * méthode appelée lorsque l'on clique sur une bille
 **********************************************/
private function billeCliquer(e:MouseEvent):void
{
        //la bille cliquer n'est plus visible sur la scène
        e.target.visible = false;

}
```

III.La boucle *do...while*

Le langage AS3 propose une autre structure répétitive, analogue à la boucle while, mais dont les instructions sont exécutées avant même d'avoir testé la condition d'exécution de la boucle. Il s'agit de la boucle do…while.

III.1.Structure

La boucle do…while se traduit par les termes faire...tant que. Cette structure s'écrit de la manière suivante :

```
do {
        instruction 1;
        instruction 2;
        instruction 3;
        ...;
        instruction n;
} while (condition);
```

La boucle *do…while* s'exécute selon les principes suivants :

- Les instructions situées à l'intérieur des accolades (boucle) sont exécutées tant que l'expression conditionnelle placée entre parenthèses () est vraie.
- Les instructions sont exécutées au moins une fois, puisque l'expression conditionnelle est examinée en fin de boucle, après exécution des instructions.

Attention, si la condition mentionnée entre parenthèses reste toujours vraie, les instructions de la boucle sont répétées à l'infini. On dit que le programme « boucle » et il plante !
Un point-virgule est placé à la fin de l'instruction *while* (expression);. N'oubliez pas d'en mettre un après le while, ou sinon votre programme plantera à la compilation.

III.2.Application

Vous allez ajouter aléatoirement sur la scène 8 carrés de couleur bleue (0x0000FF) en utilisant la boucle *do while.*

IV.La boucle *for*

IV.1.Structure

L'instruction *for* permet d'écrire des boucles dont on connaît à l'avance le nombre d'itérations (de boucles) à exécuter. Elle est équivalente à l'instruction *while* mais est plus simple à écrire.

```
for (initialisation; condition; incrémentation)
    {
    instruction 1;
    instruction 2;
    instruction 3;
    ...;
    instruction n;
    }
```

Les termes *Initialisation*, *Condition* et *Incrémentation* sont des instructions séparées obligatoirement par des points-virgules (;). Ces instructions définissent une variable, ou indice, qui contrôle le bon déroulement de la boucle. Ainsi :

- *Initialisation* permet d'initialiser la variable de la boucle (exemple : i = 0). Elle est la première instruction à être exécutée dans la boucle.
- *Condition* définit la condition à vérifier pour continuer à exécuter la boucle (exemple : i < 10). Elle est examinée avant chaque tour de boucle, y compris au premier.
- *Incrémentation* est l'instruction qui permet de modifier le résultat du test précédent en augmentant ou diminuant la valeur de la variable testée. L'incrément peut être augmenté ou diminué de N. N est appelé le « pas d'incrémentation » (exemple : i = i + 2). Cette instruction est exécutée à la fin de chaque tour de boucle.

Les boucles *for* réalisent un nombre précis de boucles dépendant de la valeur initiale, de la valeur finale et du pas d'incrémentation. Compléter les différents exemples pour voir comment ces boucles sont exécutées :

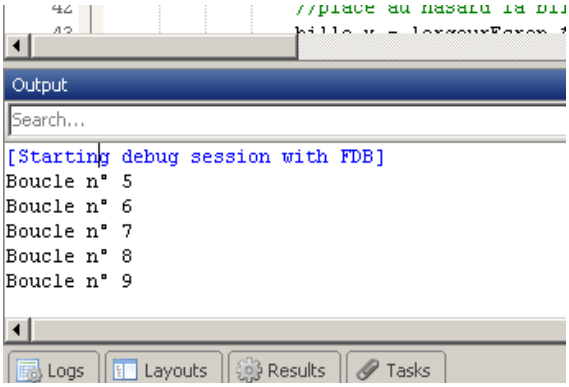
Int i	Valeur initiale	Valeur finale	Pas d'incrémentation	Nombre de boucles	Valeurs prises par i
for (i = 0; i < 5; i = i + 1)	0	4	1	5	0 1 2 3 4
for (i = 4; i <= 12; i = i + 2)					
for (i = 5; i > 0; i = i – 1)					

Remarque : Il existe d'autres boucles *for* appelées *for...in* et *for...each* . Non étudier.

IV.2.Application

Vous allez ajouter aléatoirement sur la scène 12 ellipses de couleur verte (0x00FF00) en utilisant la boucle *for.*

V.Conclusion



Les trois boucles décrites ci-dessous ont toutes pour résultat l'affichage de la fenêtre de sortie suivante :

- La boucle ***while***

```
var i:int = 5;
while (i < 10) {
        trace("Boucle n° "+ i );
        i++;
}
```

Dans la boucle while, le test de continuation de boucle s'effectue à l'entrée de la boucle. La variable i (compteur) doit donc être déclarée et initialisée correctement pour être sûr que les instructions placées à l'intérieur de la boucle soient exécutées au moins une fois.
L'instruction i++, placée à l'intérieur de la boucle, garantit que le test (i < 10) ne reste pas toujours vrai et empêche le programme de boucler à l'infini.

- La boucle ***do…while***

```
var i:int = 5;
do {
        trace("Boucle n° "+ i );
        i++;
} while (i < 10);
```

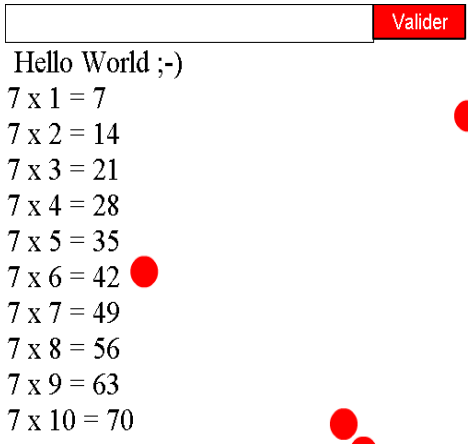
Dans la boucle do…while, le test de continuation de boucle s'effectue en sortie de boucle. Ainsi, quelle que soit la condition, nous sommes assurés que la boucle est exécutée au moins une fois.

- La boucle ***for***

```
for (var i:uint = 5; i < 10; i++) {
        trace("Boucle n° "+ i );
}
```

Dans la boucle for, la déclaration, l'initialisation, le test de continuation de boucle ainsi que l'incrémentation du compteur sont placés à l'intérieur d'une seule et même expression. La boucle *for* est utilisée quand on connaît à l'avance le nombre d'itérations (boucles) à exécuter.

VI.Exercices



6.1. Ecrire dans la zone de texte la table de multiplication par 7 en utilisant une structure for.

6.2. Ecrire dans la méthode cliqueBouton la table de multiplication du nombre rentré dans inputText en utilisant une structure while.