Créer une barre d'attente

Introduction

Dernièrement, un collègue a eu besoin de pouvoir fournir l'affichage d'une barre d'attente de façon à avoir un visuel pendant le déroulement d'un traitement.

La problématique qu'il a rencontrée a été qu'il était impossible de récupérer une quelconque information sur l'avancement de ce traitement.

Deux choix se sont alors offerts :

- Utiliser le contrôle « ProgressBar » d'AutoIt en la faisant se remplir puis se vider
- Créer notre propre control

Vous aurez surement deviné que nous avons retenu la seconde solution, la première n'étant pas intuitive à l'œil. (Donnant une impression de Rollback assez désagréable)

Remarque : L'objectif n'est pas ici de fournir une librairie UDF permettant la création d'un tel control, mais de vous expliquer le fonctionnement d'une telle chose de façon factuelle.

La création du contrôle

Pour créer un tel control, nous aurons besoin de trois objets :

- Un rectangle qui servira de délimiteur pour le control
- Un rectangle qui nous servira de fond
- Un dernier rectangle qui nous servira à réaliser la barre

Les deux premiers rectangles vont permettre de définir graphiquement le control. De façon à vous montrer la marche à suivre, voici un petit exemple qui permettra d'obtenir ceci :


```
#include <GUIConstants.au3>

Opt("GUIOnEventMode", 1)

GUICreate("Barre de progression", 220, 70)

GUISetOnEvent($GUI_EVENT_CLOSE, "_Close")

; Création du fond de notre control
GuiCtrlCreateGraphic(10, 20, 200,13)
; Définition de sa couleur (Blanc)
GUICtrlSetBkColor(-1, 0xFFFFFF)

; Création du cadre de notre control
GuiCtrlCreateGraphic(8, 18, 202,15)
; Définition de sa couleur (Noir)
GUICtrlSetColor(-1, 0x000000)

GUISetState (@SW_SHOW)

While 1
    sleep(1000)
WEnd

Func _Close()
    Exit
EndFunc
```

Ajout de la barre qui défilera

Le troisième rectangle va nous servir à créer la barre qui défilera à l'intérieur du control crée précédemment.

Le code à ajouter est simplement :

```
; Création de la barre de progression
$_bar = GuiCtrlCreateGraphic(10, 20, 49,12,$GUI_GR_PIXEL)
; Définition de sa couleur (bleu)
GUICtrlSetBkColor($_bar, 0x0000FF)
```

Le paramètre ajouté « `$GUI_GR_PIXEL` » permet simplement de rendre la barre plus esthétique. (Ce n'est qu'un avis personnel, il est également possible d'avoir un rectangle simple en enlevant ce paramètre si vous le désirez)

La différence entre cette barre et les deux autres rectangles que nous avons créés précédemment est la récupération de l'ID de celui-ci dans la variable `$_bar`.

Nous verrons plus loin que cela permettra de faire bouger notre barre.

Remarque : Les valeurs prises en exemple ici sont à titre indicatif, vous pouvez modifier les paramètres des rectangles pour obtenir une barre aux dimensions autres.

A présent, nous disposons d'un nouveau control complet nous permettant d'afficher une barre de progression.

C'est bien beau, mais s'il faut rouvrir le script à chaque fois que l'on veut modifier la coordonnée X de la barre pour la faire bouger…

Nous allons donc maintenant coder et automatiser le déplacement de notre barre.

Mise en mouvement

Pour mettre en mouvement notre barre, nous allons en premier lieu devoir définir la condition d'arrêt de celle-ci.

Remarque : Pour les besoins de l'exemple, cette condition ne sera pas fixée et la barre sera en mouvement perpétuel.

Nous allons également utiliser deux éléments :

- Une variable définissant la direction que doit prendre la barre
- Une variable définissant la position de la barre

De façon à garder une certaine simplicité, nous allons prendre les valeurs **1 et -1** pour définir les directions. Cela nous permettra d'inverser celle-ci en faisant simplement

« **$variable = $variable *-1** ».

Durant le mouvement de notre barre, nous allons vouloir pouvoir contrôler la vitesse de mouvement. Simplement, nous utiliseront la commande **Sleep(<temps>)** d'AutoIt pour donner un semblant de mouvement fluide.

Je pense qu'il sera plus aisé de comprendre le mécanisme avec un exemple concret :

```
; Entier définissant la direction de la barre
;   1: vers la droite
;  -1: vers la gauche
$stateBar = 1
; Position initiale de la barre
; Doit être initialisée à la même valeur que le fond du control
$posBar = 10

While 1 ; Condition d'arrêt de la barre d'attente

    sleep(10) ; Temps de fix de la barre

    ; Définition de la valeur suivante pour la position de la barre
    ; Si $stateBar = 1 alors $posbar = $posbar + 1
    ; Si $stateBar = -1 alors $posbar = $posbar + (-1)
    $posBar += $stateBar

    ; On change la position de la barre
    GUICtrlSetPos($_bar, $posBar, 20, 49, 12)

    ; Si la barre arrive au bout à droite ou au bout à gauche
    ; On inverse la direction
    If $posBar = 10 Or $posBar = 160 Then
        $stateBar *= -1
    EndIf

WEnd
```

Vous pourrez remarquer la valeur **160**. Celle-ci a été définie de façon à donner l'impression que la barre « rebondie » lorsqu'elle arrive à droite de notre control et changera selon la longueur que vous donnerez à celui-ci.

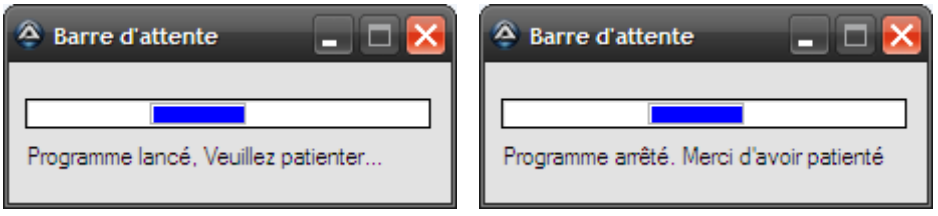
Exemple complet

De façon à vous montrer une application concrète de ce système, voici un exemple (dénué d'intérêt certes mais qui illustre bien le fonctionnement)

```
#include <GUIConstants.au3>

Opt("GUIOnEventMode", 1)

GUICreate("Barre de progression", 220, 70)

GUISetOnEvent($GUI_EVENT_CLOSE, "_Close")

; Création du fond de notre control
GuiCtrlCreateGraphic(10, 20, 200,13)
; Définition de sa couleur (Blanc)
GUICtrlSetBkColor(-1, 0xFFFFFF)

; Création du cadre de notre control
GuiCtrlCreateGraphic(8, 18, 202,15)
; Définition de sa couleur (Noir)
GUICtrlSetColor(-1, 0x000000)

; Création de la barre de progression
$_bar = GuiCtrlCreateGraphic(50, 20, 49,12,$GUI_GR_PIXEL)
; Définition de sa couleur (bleu)
GUICtrlSetBkColor($_bar, 0x0000FF)

$lblInfos = GUICtrlCreateLabel("", 9, 40, 250,15)

GUISetState (@SW_SHOW)

; Entier définissant la direction de la barre
;   1: vers la droite
;  -1: vers la gauche
$stateBar = 1
; Position initiale de la barre
; Doit être initialisée à la même valeur que le fond du control
$posBar = 10

$iID = Run(@ComSpec & " /c " & 'mspaint.exe', "", @SW_HIDE)

GUICtrlSetData($lblInfos,"Programme lancé, Veuillez patienter...")
```

```
While ($iID <> 0) ; Condition d'arrêt de la barre d'attente

    sleep(10) ; Temps de fix de la barre

    ; Définition de la valeur suivante pour la position de la barre
    ; Si $stateBar = 1 alors $posbar = $posbar + 1
    ; Si $stateBar = -1 alors $posbar = $posbar + (-1)
    $posBar += $stateBar

    ; On change la position de la barre
    GUICtrlSetPos($_bar, $posBar, 20, 49, 12)

    ; Si la barre arrive au bout à droite ou au bout à gauche
    ; On inverse la direction
    If $posBar = 10 Or $posBar = 160 Then
        $stateBar *= -1
    EndIf

    $iID = ProcessExists($iID)

WEnd

GUICtrlSetData($lblInfos,"Programme arrêté. Merci d'avoir patienté")

; Sert uniquement à garder la fenêtre ouverte
While 1
    Sleep(1000)
WEnd

Func _Close()
    Exit
EndFunc
```

Ici, nous lançons MS Paint, et la barre doit naviguer jusqu'à ce que le programme soit fermé. Dès lors que MS Paint est fermé, la barre s'arrête et le petit texte vous remercie d'avoir patienté 



Et voilà, à présent vous pouvez par exemple utiliser ce système pour faire patienter un utilisateur lors de l'installation d'un logiciel sur lequel vous n'avez aucun contrôle du déroulement !