

Une erreur est survenue lors du chargement de la version complète de ce site. Veuillez vider le cache de votre navigateur et rafraîchir cette page pour corriger cette erreur.

Les animations

EIntemporel

Les animations

Version AIO

Comme convenue après le petit sondage voici quelques TIPS pour AIO !

Vu que je suis un peu mesquin et que je ne comprends pas pourquoi la quasi-totalité des interfaces que je vois sont sans âmes, elles n'ont quasiment rien de particulier si ce n'est les effets sur les boutons "PRESSED", "SELECTED" & "HIGHLIGHT"... Et encore si vous utilisez un template, vous n'avez même pas besoin de vous en soucier... BREF !

Bah je vais vous apprendre à créer tout un tas d'animations pour que ça soit beaucoup plus festifs, pour que vos interfaces soient exceptionnelles et que les joueurs qui vont les découvrir se sentent submergés par l'émotion !

J'espère qu'après ce petit tutoriel de rien du tout, vous allez vous amuser à animer vos interfaces et qu'on pourra partager encore plus de TIPS autour des interfaces et de l'AIO, parce que pour le coup j'ai l'impression d'être tout seul à utiliser des animations...

Les différentes Animations qu'il existe

Bon ce n'est pas compliqué, il en existe 5 (plus les animations customs), je vais les énumérer dans l'ordre dans lesquelles je vais les traiter ! Il faut aussi savoir que l'animation appelée Path, qui met en lien d'autres animations pour n'en former qu'une et une seule est à proprement parler PAS une animation. Mais nous verront ça plus tard dans un autre tutoriel !

- o ROTATION
- o SCALE
- o ALPHA
- o TRANSLATION
- o PATH (ON NE LE VERRA PAS ICI)

Comprendre le système d'Animation

Ici ce n'est clairement pas compliqué du tout, les développeurs chez Blizzard ont vraiment fait un joli truc même pour la version 3.3.5, il faut savoir que le système d'animation repose sur **2 UIObjects** vraiment très simple de compréhension puisqu'il s'agit de :

- o ANIMATIONGROUP
- o ANIMATION

Bon en gros si vous n'avez pas compris, l'**UIObject AnimationGroup** contient les **UIObjects Animation**, une animation par exemple de rotation est donc défini dans un **UIObject Animation** et lui-même est contenu dans un **UIObject AnimationGroup** (celui-ci peut contenir une multitude de UIObject Animation).

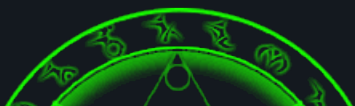
Schéma explicatif

ROTATION -> UIObject Animation -> UIObject AnimationGroup -> Action

Toutes les caractéristiques d'une animation sont définies dans un **UIObject Animation** et toutes les actions (Play, Pause, etc...) sont utilisées sur les **UIObjects AnimationGroup**.


La base de notre script

Bon vu qu'il faut bien commencer avec quelque chose de concret, pour toutes nos animations on va utiliser la texture suivante :





Et pour notre script de base (je ne vais pas expliquer les bases ici, donc je vous passe mon script de base et puis voilà), j'ai rapidement codé ceci (lol vous n'avez quand même pas cru pouvoir copier/coller mon script ? De toute façon même si c'est que de réécrire, ça aide pour rentrer dans la tête) :

Scripts >  tutorial.lua > ...

```
1
2  local AIO = AIO or require("AIO")
3  if AIO.AddAddon() then
4      return
5  end
6
7  -- Définition de la table TUTORIEL qui contiendra toute l'interface du tutorial
8  local TUTORIEL = {}
9
10 TUTORIEL = CreateFrame("Frame", TUTORIEL, UIParent, "UIPanelDialogTemplate")
11     TUTORIEL:SetSize(300,300)
12     TUTORIEL:SetPoint("CENTER")
13     TUTORIEL:SetToplevel(true)
14     TUTORIEL:Show()
15
16     TUTORIEL.obj = TUTORIEL:CreateTexture()
17     TUTORIEL.obj:SetTexture("Interface/SpellShadow/Spell-Shadow-Acceptable")
18     TUTORIEL.obj:SetSize(150,150)
19     TUTORIEL.obj:SetPoint("CENTER")
20
```

Ce qui donne lorsque l'on réinitialise en jeu :





Ouai je sais c'est déjà vachement sympa comme ça !

Partie 1 : Rotation

Ici c'est très simple (comme plus tard d'ailleurs), il faut tout d'abord créer notre UIObject AnimationGroup, puis notre UIObject Animation qui va contenir la définition de la rotation, du coup, hop c'est partie.... 2 lignes de code plus tard on obtient :

```
--AnimationGroup
TUTORIEL.AnimationGroup = TUTORIEL.CreateAnimationGroup()
--Animation
TUTORIEL.AnimationRotation = TUTORIEL.AnimationGroup.CreateAnimation("rotation")
```

Et oui c'est aussi simple que ça pour la partie AnimationGroup, par contre pour l'UIObject Animation ce n'est pas encore fini, il faut qu'on définisse de combien on souhaite effectuer notre rotation, et pendant combien de temps, ce qui nous donne (j'effectue une rotation de 360 degrés sur une période de 10 sec)

```
TUTORIEL.AnimationRotation:SetDegrees(360)
TUTORIEL.AnimationRotation:SetDuration(10)
```

C'est encore une fois aussi simple que ça, donc HOP retournez en jeu, fait un /reload eluna, et regardez votre traa.. Bah quoi il ne se passe rien ? En fait c'est normal, ici on a juste défini notre UIObject Animation, mais on ne lui a jamais dit de s'exécuter, du coup c'est encore aussi simple, là ça se passe niveau AnimationGroup et on écrit par la suite :

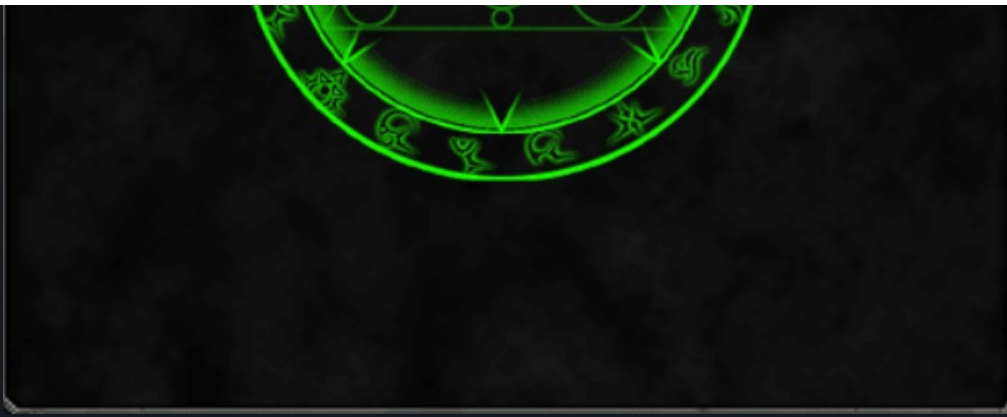
```
TUTORIEL.AnimationGroup:Play()
```

Et là... c'est le drame, vous venez de découvrir qu'une animation agit sur toutes les frames enfants ! Regardez-moi ce carnage ! Bon vous allez vous demander comment est-ce possible ? Et comment résoudre ce problème ? C'est simple, vous avez défini l'UIObject de l'AnimationGroup, directement sur la frame TUTORIEL, et pas sur la texture TUTORIEL.obj, du coup après une petite modification

```
--AnimationGroup
TUTORIEL.AnimationGroup(TUTORIEL.obj):AnimationGroup()
```

Il n'y a désormais que la texture qui est en mouvement





Pour aller encore plus loin dans cette animation, on peut définir une boucle à notre AnimationGroup, qui va permettre à notre Animation de tourner sans arrêt, (cette fonction est utilisable sur toutes les autres animations), et c'est la fonction SetLooping, simplement utilisable comme suit, vous pouvez la mettre après le lancement de l'animation

```
TUTORIEL.AnimationGroup:SetLooping("BOUNCE")
--TUTORIEL.AnimationGroup:SetLooping("REPEAT")
--TUTORIEL.AnimationGroup:SetLooping("NONE")
```

BOUNCE : permet de faire rebondir l'animation, effectue le chemin inverse

REPEAT : permet de répéter l'animation en boucle, effectue le même cycle sans cesse

NONE : Aucun effet de boucle

Partie 2 : Scale

Ici, en reprenant le code de base qui est celui-ci :

```
Scripts > tutorial.lua > ...
1
2  local AIO = AIO or require("AIO")
3  if AIO.AddAddon() then
4      return
5  end
6
7  -- Définition de La table TUTORIEL qui contiendra toute l'interface du tutorial
8  local TUTORIEL = {}
9
10 TUTORIEL = CreateFrame("Frame", TUTORIEL, UIParent, "UIPanelDialogTemplate")
11     TUTORIEL:SetSize(300,300)
12     TUTORIEL:SetPoint("CENTER")
13     TUTORIEL:SetToplevel(true)
14     TUTORIEL:Show()
15
16     TUTORIEL.obj = TUTORIEL:CreateTexture()
17     TUTORIEL.obj:SetTexture("Interface/SpellShadow/Spell-Shadow-Acceptable")
18     TUTORIEL.obj:SetSize(150,150)
19     TUTORIEL.obj:SetPoint("CENTER")
20
```

On remet en place notre UIObject AnimationGroup, et notre UIObject Animation qui va détenir cette fois-ci la définition de notre Scale, on ajoute donc :

```
--AnimationGroup
TUTORIEL.AnimationGroup = TUTORIEL.obj:CreateAnimationGroup()
--Animation
```

```
TUTORIEL.AnimationScale = TUTORIEL.AnimationGroup:CreateAnimation("scale")
```

Pour l'animation de Scale il y a donc aussi des paramètres à mettre en place, comme tout à l'heure il y a un paramètre de temps, puisqu'il s'agit d'une animation, et d'un paramètre de taille, qui sont :

```
TUTORIEL.AnimationScale:SetScale(2, 2)
```

```
TUTORIEL.AnimationScale:SetDuration(1)
```

Je ne vous explique plus qu'est-ce que c'est que SetDuration(), par contre SetScale(x,y) permet simplement d'indiquer le facteur (donc multiplicateur), par lequel on va multiplier la taille de notre objet, la valeur X permettra de multiplier la largeur de l'objet et la valeur Y sa hauteur, par exemple pour grossier de 2 fois la taille de notre objet on indiquera alors SetScale(2,2), et si on souhaite le diviser de 2 fois, ça sera alors SetScale(0.5, 0.5) ($1/2 = 0.5$), tout simplement, et comme tout à l'heure on oublie pas de lancer notre animation depuis AnimationGroup, et de définir la boucle voulu (ici pour une animation de scale, la boucle BOUNCE est largement utiliser pour simuler un "cœur battant"), ce qui nous donne :

```
TUTORIEL.AnimationGroup:Play()
```

```
TUTORIEL.AnimationGroup:SetLooping("BOUNCE")
```

Une fois en jeu le rendu est le suivant :



Partie 3 : Alpha

Je pense que vous avez fini par comprendre le sens que prend ce tutoriel du coup on passe directement à l'essentiel, ici l'animation Alpha, va prendre comme tout à l'heure un paramètre de temps définis par SetDuration(), et un paramètre de modification de l'Alpha, définis par SetChange(x), avec X qui prend en valeur la modification voulu sur l'alpha de notre objet, ce qui implique que (animation est portée sur l'objet nommé ici test):

Anim:SetChange(-0.25) équivaut à test:SetAlpha(test:GetAlpha()+(-0.25)) (modification effectuée progressivement en fonction de la durée indiqué pour notre animation)

ALPHA ne prend des valeurs compris qu'entre 0 et 1 inclus, le client se charge de tout si jamais vous mettez des valeurs non compris entre 0 et 1 (il les rapporteras à la borne la plus proche)

```
--AnimationGroup
```

```
TUTORIEL.AnimationGroup = TUTORIEL.obj:CreateAnimationGroup()
```

```
--Animation
```



```
TUTORIEL.AnimationAlpha = TUTORIEL.AnimationGroup.CreateAnimation("alpha")

TUTORIEL.AnimationAlpha:SetChange(-0.8)
TUTORIEL.AnimationAlpha:SetDuration(1)

TUTORIEL.AnimationGroup:Play()
TUTORIEL.AnimationGroup:SetLooping("BOUNCE")
```

Le rendu en jeu est le suivant :



Partie 4 : Translation

Pour l'animation de translation, celle-ci prendre comme tout à l'heure un paramètre de temps définis par `SetDuration()`, et un paramètre de modification de coordonnées, définis par `SetOffset(x,y)`, avec X et Y des valeurs de déplacement,

Par exemple `SetOffset(20,0)` déplacera l'objet de 20 pixel sur l'axe des abscisses par rapport à son emplacement initial et de 0 sur l'axe des ordonnées



Pour moi ça donne ceci :

```
--AnimationGroup
TUTORIEL.AnimationGroup = TUTORIEL.obj:CreateAnimationGroup()
--Animation
TUTORIEL.AnimationTranslation = TUTORIEL.AnimationGroup:CreateAnimation("translation")

TUTORIEL.AnimationTranslation:SetOffset(25, 25)
TUTORIEL.AnimationTranslation:SetDuration(1)

TUTORIEL.AnimationGroup:Play()
TUTORIEL.AnimationGroup:SetLooping("BOUNCE")
```

Et le rendu en jeu :



Partie BONUS : La combinaisons des animations

Oui vous avez bien lu, on peut combiner des animations par exemple sur mon exemple j'ai décider de combiner les animations : **Rotation**
– **Alpha** – **Translation**

Je n'ai simplement qu'à les ajouter les unes à la suite des autres comme ceci :

```
--AnimationGroup
TUTORIEL.AnimationGroup = TUTORIEL.obj:CreateAnimationGroup()
    local GLOBAL_DURATION = 2
--Animation
TUTORIEL.AnimationRotation = TUTORIEL.AnimationGroup:CreateAnimation("rotation")
    TUTORIEL.AnimationRotation:SetDegrees(360)
    TUTORIEL.AnimationRotation:SetDuration(GLOBAL_DURATION)

TUTORIEL.AnimationAlpha = TUTORIEL.AnimationGroup:CreateAnimation("alpha")
    TUTORIEL.AnimationAlpha:SetChange(-0.8)
    TUTORIEL.AnimationAlpha:SetDuration(GLOBAL_DURATION)

TUTORIEL.AnimationTranslation = TUTORIEL.AnimationGroup:CreateAnimation("translation")
    TUTORIEL.AnimationTranslation:SetOffset(25, 25)
    TUTORIEL.AnimationTranslation:SetDuration(GLOBAL_DURATION)

TUTORIEL.AnimationGroup:Play()
TUTORIEL.AnimationGroup:SetLooping("BOUNCE")
```

Ici comme vous l'avez vu j'ai défini des mêmes durées pour toutes mes animations parce que vu que je les fait "rebondir", le script attendra que toutes les animations ai fini de s'exécuter pour justement rebondir, en jeu cela me donne :



Si vous voulez séparer les animations pour leurs appliquer des durées différentes vous devais alors créer plusieurs UIObject AnimationGroup (sur la même texture) comme suit :

```
--AnimationGroup1
TUTORIEL.AnimationGroup1 = TUTORIEL.obj:CreateAnimationGroup()
    local GLOBAL_DURATION_1 = 2
```



```

TUTORIEL.AnimationAlpha = TUTORIEL.AnimationGroup1:CreateAnimation("alpha")
TUTORIEL.AnimationAlpha:SetChange(-0.8)
TUTORIEL.AnimationAlpha:SetDuration(GLOBAL_DURATION_1)

TUTORIEL.AnimationTranslation = TUTORIEL.AnimationGroup1:CreateAnimation("translation")
TUTORIEL.AnimationTranslation:SetOffset(25, 25)
TUTORIEL.AnimationTranslation:SetDuration(GLOBAL_DURATION_1)

--AnimationGroup2
TUTORIEL.AnimationGroup2 = TUTORIEL.obj:CreateAnimationGroup()
local GLOBAL_DURATION_2 = 10

TUTORIEL.AnimationRotation = TUTORIEL.AnimationGroup2:CreateAnimation("rotation")
TUTORIEL.AnimationRotation:SetDegrees(360)
TUTORIEL.AnimationRotation:SetDuration(GLOBAL_DURATION_2)

TUTORIEL.AnimationGroup1:Play()
TUTORIEL.AnimationGroup1:SetLooping("BOUNCE")

TUTORIEL.AnimationGroup2:Play()
TUTORIEL.AnimationGroup2:SetLooping("BOUNCE")

```

Et le rendu en jeu :



Comme vous l'avez peut-être remarqué, les animations de Translation et Alpha s'effectue bien en 2 sec, et celle de Rotation s'effectue quand à elle en 10 sec, ce qui rend l'animation global plus jolie à voir.

Conclusion

J'espère que c'est petits tips pour les animations vous ont plu, si jamais je me rend compte que j'ai oublié de parler de quelque chose je vous le mentionnerais !

Bon développement à vous tous !

EIntemporel

iThorgrim

OUI ! Merci pour ce guide / tutoriel ! ☐

EIntemporel

En plus y'a plein de gif et d'image exprès pour que ça soit jolie!

iThorgrim

↩ **EIntemporel** Hésite pas à utiliser des titres plus grands quand tu mets des images, comme ça tes titres on les vois un peu plus :D

Parce que du coup il passe un peu inaperçu du fait que tu as pas mal d'images ^^ (et surtout du fait que l'éditeur est pas ouf pour le moment)

Torvahal

Merci pour le partage ☐

RaverKai

Merci beaucoup pour le partage de la connaissance, mes sincères respects ☐

noc

Plus qu'un tutoriel, c'est une base qui nous permettra de faire bien des choses, un grand merci pour tout ce travail.
