Les Expressions sur Adobe After Effects

Formation de DuDuF
http://formations.duduf.fr
duduf@duduf.com

I - Objets, Valeurs

Que peut-on manier avec les expressions ? Quels sont les types de valeurs ? Comment les utilise-t-on ?

En pratique : manier les outils de création d'expressions, lier des propriétés, créer des contrôleurs.

II - Opérations

Variables, Opérations mathématiques, additionner, multiplier, etc.

En pratique : Combiner plusieurs propriétés, utiliser les fonctions mathématiques...

III - Conditions

If, then, else

En pratique : automatiser des évènements, insérer des conditions.

IV - Fonctions

Fonctions prédéfinies, fonctions créées
En pratique : Utiliser les fonctions After Effects, créer ses propres fonctions.

Ce document n'est qu'un résumé, un pense-bête, de la formation dispensée par Duduf. Il réunit les principaux points théoriques à retenir sans être exhaustif, et de nombreuses remarques, astuces pratiques et exemples concrets n'y sont pas présents.

I - Objets, Valeurs

Adobe After Effects a un mode de fonctionnement très proche du langage JavaScript, couramment utilisé sur internet. C'est un langage de programmation « orienté objet ». L'utilisation des expressions est intimement lié à ce mode de fonctionnement. Commençons donc par y jeter un oeil.

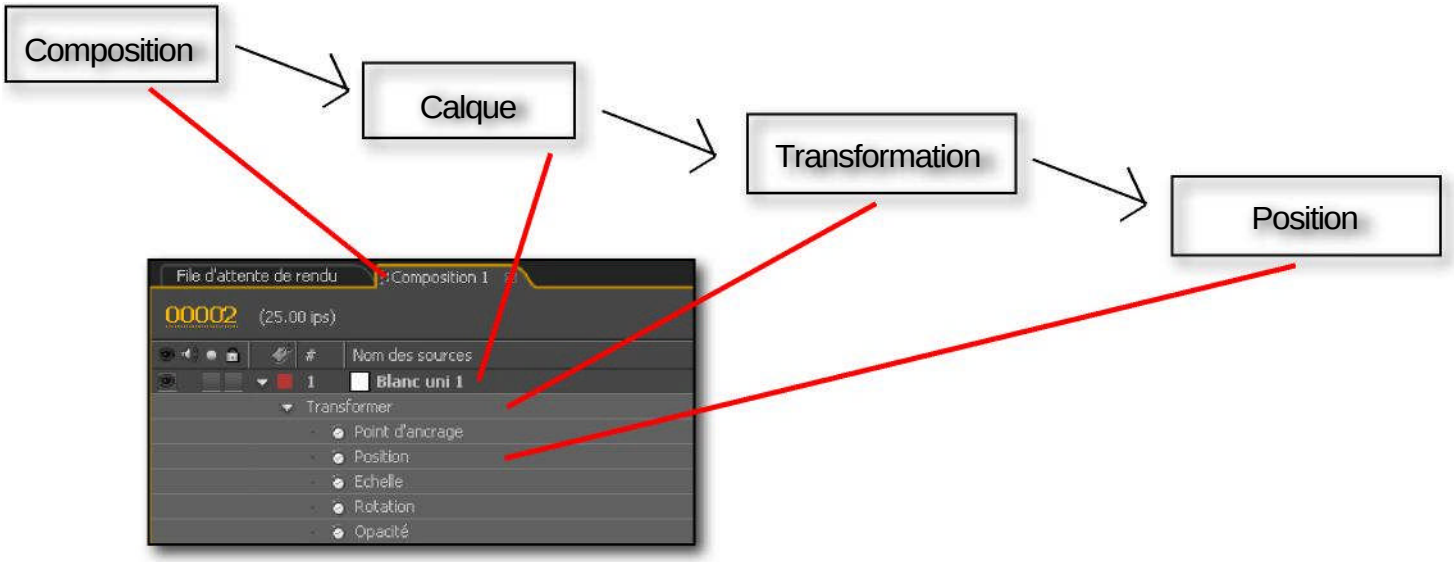
Un objet, c'est quoi ?

Pour comprendre, prenons des exemples d'objets : composition, calque, échelle, rotation, flou... Eh oui, un objet, c'est tout ce qu'on peut manipuler dans une composition, et donc aussi avec les expressions.

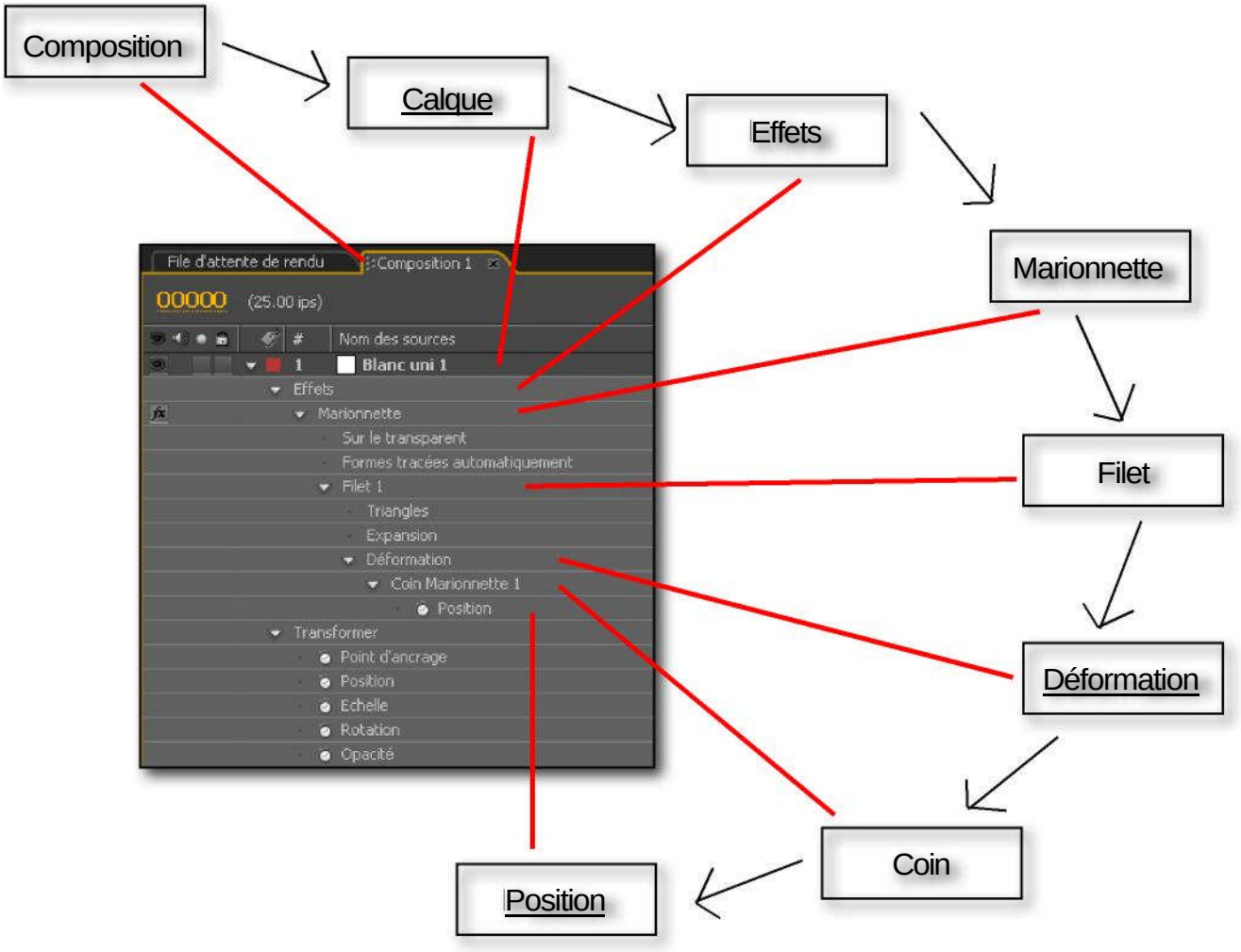
Mais encore ?

Tous les objets sont classés de façon hiérarchique : Le grade le plus élevé est la composition, le plus bas est le paramètre d'effet.
On peut dire que les objets sont imbriqués les uns dans les autres, un peu comme des poupées russes.

Voici un exemple de quatre objets, et la hiérarchie entre eux :



Un autre exemple plus complexe :



Que faire de tous ces objets ?

Nous allons mettre des valeurs dedans ! L'objet le plus bas dans la hiérarchie est celui qui contient une valeur (dans l'exemple, la position). La plupart du temps, cette valeur est animable (repérable au chronomètre juste à côté sur After Effects). Et la grande majorité (mais pas toutes) de ces valeurs peut être contrôlée par une expression !

Un objet ne peut contenir qu'une valeur.

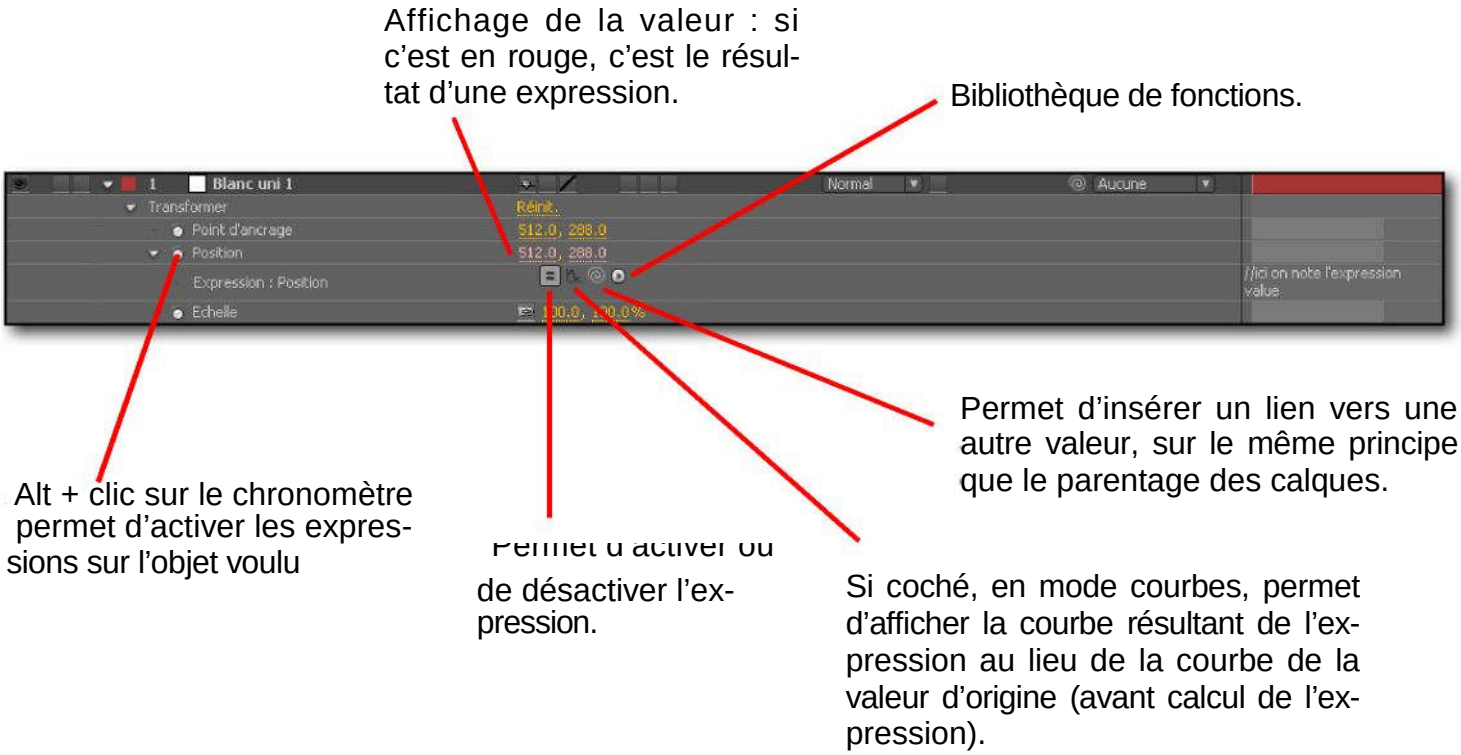
Mais une position, ce ne sont pas DEUX valeurs ?

Eh non ! Il y a les valeurs simples, comme la valeur d'un flou ou d'un angle, mais certains objets font appel au **tableau**. C'est un type de valeur qui contient en fait plusieurs autres valeurs (position X, Y et Z pour la position par exemple). La plupart des objets font appel soit à une valeur simple, soit à un tableau de valeurs, mais quelques objets utilisent des valeurs encore différentes, comme les tracés de masque...

Bon, et les expressions dans tout ça ?

Les expressions sont donc un moyen de contrôler ces valeurs. En fait, une expression, c'est une commande, une opération mathématique le plus souvent, dont le résultat donne la valeur à appliquer à l'objet.

Regardons donc comment manier ces expressions !



Essayons de créer une expression !

La plus simple de toutes : la valeur.

- Sur un calque, ouvrez les propriétés de transformation, et alt+clic sur le chronomètre de l'opacité.
- Tapez « *20* », par exemple, dans le champ de l'expression.

On remarque deux choses : l'affichage de la valeur de l'opacité est devenu rouge, pour indiquer qu'elle est le résultat d'une expression, et si on modifie cette valeur en cliquant dessus, elle se remet toujours à 20. En effet, quoiqu'il arrive, le résultat de l'expression est une constante, 20.

- Cliquez sur le bouton =. L'affichage de la valeur change et redevient jaune. Ce n'est plus le résultat de l'expression, mais la valeur définie pour le calque.

Et la position ?

Essayons !

- Sur un calque, activez les expressions sur la position.
- Tapez « *20* ».

Aïe, ça ne marche pas, After me parle de dimensions...

Souvenez-vous, la position prend un tableau de valeurs !
C'est pour ça qu'After Effects demande un résultat de dimension 2 (un tableau à deux valeurs) et pas 1 !

- Entrez maintenant « *[20,20]* ».

Maintenant ça fonctionne. Vous venez de découvrir comment entrer un tableau dans une expression, entre crochets, les valeurs séparées par des virgules.

Mais, et les valeurs à virgule alors ? comment les écrire, si la virgule sert de séparateur entre deux valeurs ?

Nous arrivons à un point très important : les développeurs de After Effects sont américains ! Les expressions sont donc en anglais et en anglais, les séparateurs sont des virgules, et les nombres décimaux s'écrivent avec un point ! Il va falloir s'y faire...

Une constante c'est bien, mais le but c'est quand même d'animer...

Créons notre première expression dont le résultat change dans le temps ! Oui, c'est juste ça une animation, un résultat qui change en fonction de l'instant.
La plus simple des expressions animées, c'est le lien entre deux valeurs.

- Créez deux solides, dépliez leurs transformations.
- Activez les expressions sur l'opacité du premier, et alors que le champ de saisie est actif, utilisez le lasso pour pointer la valeur de l'opacité du deuxième puis validez.

L'expression ressemble alors à « *thisComp.layer("Nom du Calque").transform.opacity* ».
On y retrouve la hiérarchie des objets, et comment on les désigne :
La composition « thisComp » qui montre la composition actuelle.
Le calque « *layer("Nom du Calque")* » que l'on peut remplacer par « layer(X) » où X représente son index.
La transformation, et l'opacité (toujours en anglais).

Maintenant, si on anime l'opacité du deuxième calque, le premier suit exactement la même animation !

Exerçons nous pour terminer ce chapitre ! Créons nos premières expressions !

- Sur un solide noir, appliquez un effet «génération/lumière parasite».
- Créez un objet nul.
- Utilisez une expression pour lier le «Centre de la source lumineuse» à la position de l'objet nul.

C'est une utilisation très pratique des expressions pour animer certains effets ; dans ce cas particulier il est nettement plus facile d'animer la position d'un calque que le centre d'un effet !

- Sur le solide noir, ajoutez un effet «netteté/flou radial».
- Liez le centre du flou à la position de l'objet nul avec une expression.
- Ajoutez sur l'objet nul un effet «options pour expressions/paramètre glissière».
- Liez l'intensité du flou au curseur du paramètre glissière.
- Liez l'intensité de la lumière parasite au paramètre glissière.

Ici nous créons un contrôleur : un objet très utile avec les expressions, pour contrôler plusieurs effets en même temps, pour donner des paramètres à des expressions... Et qui permet par ailleurs de rassembler tous les paramètres à animer sur le même calque (ici, l'objet nul).
Un autre exemple de l'utilisation d'un tel contrôleur, c'est de séparer les différentes valeurs d'un objet à plusieurs dimensions, comme la position :

- Créez un solide, et ajoutez-y deux paramètres glissière.
- Activez les expressions sur la position, tapez un premier crochet «[» puis avec le lasso liez le curseur du premier paramètre glissière, ajoutez une virgule «,», liez le deuxième paramètre glissière, et enfin fermez les crochets «]».

Nous venons de recréer l'effet «Séparer la position XYZ» ! Aucun intérêt puisqu'il existe déjà, mais vous savez maintenant le faire aussi sur n'importe qu'elle position, le centre d'un effet, un point de marionnette, etc. !

Pensez déjà à toutes les possibilités des expressions en combinant ces simples bases !

II - Opérations

Nous savons maintenant assigner une valeur et récupérer la valeur d'un autre objet. C'est déjà pratique, mais encore loin des possibilités des expressions. Il sera encore bien plus utile de pouvoir associer plusieurs valeurs et réaliser des calculs. Pour cela nous allons utiliser des opérateurs, pour réaliser des opérations entre différentes opérandes.

Un opérateur, c'est le «signe» de l'opération (+, -, =, etc.). Il y en a plein, pour faire plein de choses.

Une opérande, c'est l'élément sur lequel travaille l'opérateur (dans 2+2=4, les 2 et le 4 sont les opérandes).

Commençons par les maths, niveau école primaire.

Quatre opérations, quatre opérateurs : addition «+», soustraction «-», division «/», multiplication «*». Rien de bien sorcier... Une expression «*20+30*» donnera 50, «*20*30*» donnera 600.

Et puisque c'est trop simple, ajoutons un cinquième opérateur, le modulo «%». Il retourne le reste de la division entière des deux opérandes. «*10%8*» donnera 2. «*30%8*» donnera 6.

Si on écrit une opération à plus de deux opérandes, c'est comme sur le papier, il y a des priorités : «*», «/», «%» auront la priorité sur «+» et «-». Et bien sûr, si ça ne nous plait pas, on peut ajouter des parenthèses. «*2+2*4*» donnera 10 et «*(2+2)*4*» donnera 16. Toujours rien de bien sorcier, mais revenons à du plus concret...

Utiliser des expressions pour y mettre des valeurs que l'on peut trouver avec une calculatrice, ce n'est pas très utile. Mais remplaçons nos opérandes numériques par des liens vers les valeurs d'autres objets...

- Créez trois solides, animez différemment l'opacité de deux d'entre eux, et ajoutez une expression sur le troisième.
- Liez la valeur d'opacité du premier solide, ajoutez «+», puis liez la valeur d'opacité du deuxième solide. Vous obtenez une expression du type :
«*thisComp.layer("calque1").transform.opacity + thisComp.layer("calque2").transform.opacity*»
(on peut ajouter des espaces pour y voir plus clair, les espaces n'influencent aucunement les expressions).

La valeur d'opacité du troisième calque est donc la somme des deux autres, à n'importe quel moment dans le temps. Là ça commence à devenir intéressant.

Dis donc, ça risque de devenir illisible et dur à organiser si on a beaucoup d'opérandes !

Dans les cas où on lie beaucoup d'objets, où les opérandes sont nombreuses, et à plus forte raison dans les cas où on utilise plusieurs fois la même opérande, il y a un outil très pratique : la variable ! Elle permet d'assigner une opérande à un nom de notre choix, plus simple. Par exemple au lieu de mettre «*thisComp.layer("calque1").transform.opacity*», on va pouvoir utiliser «calque1Opacité» par exemple. Pour créer (on dit déclarer en programmation) une variable, c'est tout bête :

- Reprenez les trois solides de tout à l'heure. Celui où on a mis l'expression, vous allez effacer l'ancienne et créer celle ci :
«*calque1Opacité = thisComp.layer("calque1").transform.opacity;*
calque2Opacité = thisComp.layer("calque2").transform.opacity;
calque1Opacité + calque2Opacité»

Le résultat est exactement le même que tout à l'heure, mais la dernière opération est bien plus facile à lire. C'est une méthode très utile lors de la création d'expressions plus compliquées.

Notons au passage la syntaxe utilisée par les expressions : un point-virgule à la fin de chaque ligne pour séparer les opérations, sauf à la dernière ligne, celle dont le résultat doit servir de valeur pour l'objet dans lequel l'expression est appliquée.

Et aussi, ce nouvel opérateur «=», qui permet d'affecter à l'opérande de gauche la valeur de celle de droite. (attention, ce n'est pas un opérateur d'égalité, mais un opérateur d'affectation). Comme tous les opérateurs, il subit les priorités. Il vient après «+» et «-» dans l'ordre des priorités.

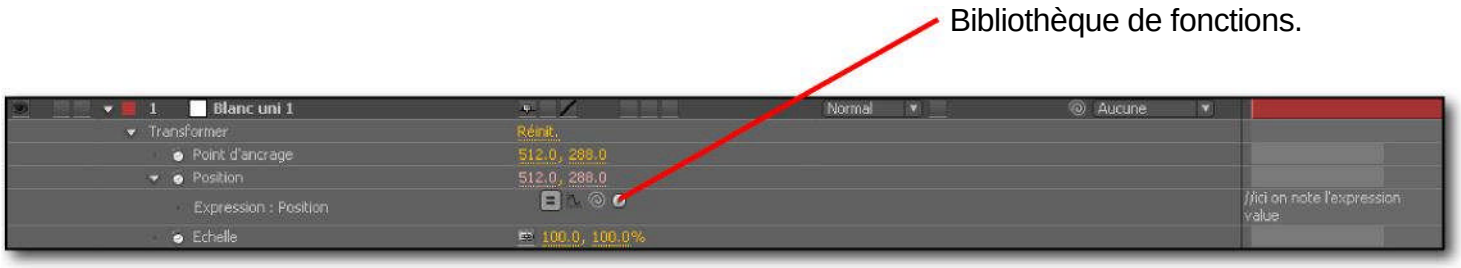
Maintenant, nous connaissons donc les 6 opérateurs de calcul (en comptant l'affectation), ajoutons-y 5 autres opérateurs pas indispensables mais qui peuvent servir à simplifier nos expressions : les opérateurs d'affectation. Exemple : dans certains cas, vous aurez besoin de modifier une variable, d'y additionner ou soustraire une valeur, avant de la réutiliser. Vous pourriez le faire de la façon suivante : «*variableA = variableA + 23*». L'opérateur d'affectation simplifie cette expression de la façon suivante : «*variableA += 23*». Il y en a donc 5 : «+=», «-=», «*=», «/=», «%=». Ils ont tous les 5 la même priorité que l'opérateur d'affectation «=».

Il y a encore deux opérateurs, très utiles en programmation, mais peu utilisés par les expressions : les opérateurs d'incrémentation : «++» et «--» qui additionnent ou soustrayent une unité à l'opérande placée juste à gauche. Exemple : «*variableA++*» est équivalent à «*variableA += 1*» ou encore à «*variableA = variableA + 1*». Ces deux opérateurs ont la priorité sur «*», «/» et «%», ils sont donc ceux dont la priorité est la plus haute.

Il est temps d'entrer au collège !

Très vite, vous aurez besoin d'effectuer des calculs un peu plus complexes... Récupérer des valeurs absolues, calculer des puissances, des racines carrées, travailler des angles, etc. After Effects dispose d'un certain nombre de fonctions prêtes à faire ça pour vous !

Prenons un tout petit peu d'avance sur le dernier chapitre et regardons où trouver ces fonctions, et comment les utiliser :



Dans l'onglet «JavaScript Math» vous y trouverez par exemple :

le cosinus : «*Math.cos(valeur)*»
le cosinus inversé : «*Math.acos(valeur)*»
la puissance : «*Math.pow(valeur, puissance)*»
la valeur absolue : «*Math.abs(valeur)*»
ou encore, l'arrondi supérieur : «*Math.ceil(valeur)*»

Et dans «Other Math», pour la conversion de radian en degrés : «*radiansToDegrees(valeur)*» ou le contraire «*degreesToRadians(valeur)*».

Attention aux majuscules/minuscules, elles sont importantes dans les expressions !

Exerçons nous pour terminer ce chapitre !

- Créez trois solides (d'une taille plus petite que celle de la compo, pour bien voir leur position)
- Animez la position des deux premiers, et ajoutez l'expression suivante en position sur le troisième :

«*positionA = thisCompiayer("calque1").transform.position;*
positionB = thisCompiayer("calque2").transform.position;
(positionA + positionB) / 2»

Vous venez de faire la moyenne des positions, le calque 3 est toujours a mi-chemin des deux autres. Vous commencez à trouver des applications de plus en plus utiles aux expressions... Essayez aussi avec des coefficients !

«*positionA = thisCompiayer("calque1").transform.position;*
positionB = thisCompiayer("calque2").transform.position;
*(positionA*2 + positionB) / 3*»

Essayez avec encore plus de calques, faites la moyenne de 3, 4 ou même 5 positions... Au passage notons que nous travaillons sur des tableaux de valeurs à 2 ou même 3 dimensions (puisque ce sont des positions), exactement de la même façon que si c'étaient des valeurs simples. After Effects se charge du calcul vectoriel pour nous !

Regardons maintenant une autre expression :

- Créez un solide de 200pixels sur 200 (dans une compo plus grande...) - Ajoutez cette expression dans la rotation :
«*centre = thisLayer.position;*
R = 100;
radiansToDegrees(centre[0]/R)»

Vous venez d'inventer la roue ! Cette expression est presque la même que celle que crée l'outil «roue» de Duik (http://ik.duduf.fr). Elle automatise la rotation du calque en fonction de son déplacement horizontal pour faire «rouler» le calque. (évidemment, si vous mettez un masque en cercle ce sera plus parlant).

Analysons l'expression :

«*thisLayer.position*» est un abrégé de «*thisComp.thisLayer.transform.position*». D'ailleurs, on aurait même pu mettre simplement «*position*» tout court, mais on n'aurait pas découvert le terme thisLayer en passant ;-). La première ligne assigne donc à la variable nommée «centre» la valeur de la position du calque courant.

«*R = 100*» assigne bètement la valeur 100 à une variable «R». Pourquoi 100 ? si le calque fait 200 pixels de large, c'est tout simplement le rayon de la roue...

«*centre[0]*», encore quelque chose de nouveau : ça permet de récupérer la première valeur du tableau «centre» (centre est une variable qui contient une position, donc un tableau de 2 ou 3 valeurs), donc la valeur de la position en X, horizontale. «*centre[1]*», la valeur en Y, «*centre[2]*», la valeur en Z.

Si on se souvient des cours de maths du collège, en radians : 2π*R c'est la circonférence du cercle, donc la distance parcourue quand il fait un tour complet. D'ailleurs, 2π, en radians, c'est l'angle correspondant à un tour complet (360, en degrés). Doooooonc (vous me suivez toujours ?) : Angle*Rayon=Distance (en radians, toujours). Et donc on a Angle=Distance/rayon. Bah voilà : «*centre[0]/R*» qu'on n'a plus qu'à convertir en degrés, puisque After Effects travaille en degrés, «*radiansToDegrees(centre[0]/R)*» et le tour est joué !

Pour faire monter la roue à la verticale, changez pour «*centre[1]/R» !*

Ces deux exemples montrent des expressions pour l'animation de transformations, mais nous avons maintenant un attirail très complet pour créer des expressions pour contrôler précisément tous les effets, que ce soit leurs intensités, leurs positions, etc.

III - Conditions

Maintenant que nous savons manipuler toutes sortes de valeurs, les lier, les associer, calculer, il ne reste plus que : «A partir de là, change de sens» ou encore «Si je te met à droite, devient transparent». C'est à dire les conditions, «if(){} else{}»

Une seule condition.

Le principe est simple : on veut qu'un calque devienne transparent si il passe sur la droite de la composition.

La syntaxe est la suivante :
if (condition) {ce qui se passe si la condition est remplie}
else {ce qui se passe si la condition n'est pas remplie}

On mettra donc une expression dans l'opacité :
«*if (transform.position[0] < thisComp.width/2) {100}*
else {50}»

Notons en passant «*thisComp.width*» : la largeur de la compo, en pixels, et le nouvel opérateur inférieur «<» dont la priorité passe après «*», «/», «%».

Nous allons d'ailleurs avoir besoin de toute une série d'opérateurs de comparaison pour établir nos conditions. Les voici :
inférieur «<»
supérieur «>»
inférieur ou égal «<=»
supérieur ou égal «>=»
égalité «==» (à ne pas confondre avec affectation «=»)
différence «!=»

Dans les effets/options pour expressions, le paramètre case permettra de créer un bouton pour allumer ou éteindre un calque par exemple :
«*if (effect("Paramètre case")("Case")==1) {100}*
else {50}»

La case vaut 1 si elle est cochée, 0 si elle est décochée...

Et si j'ai plusieurs conditions à poser ?

Essayons de réunir les deux cas précédents : si la case est cochée, alors si le calque est sur la droite, allumons le. Sinon, il reste éteint. Dis comme ça, on a envie de créer l'expression suivante :
«*if (effect("Paramètre case")("Case")==1) {*
if (transform.position[0] < thisComp.width/2) {100}
}
else {50}»

Non ce n'est pas un piège, ça fonctionne. Mais il y a plus simple... Pourquoi ne pas se dire simplement «si la case est cochée ET si le calque est sur la droite, allumons le, sinon il reste éteint.»
«*if (effect(«Paramètre case»)(«Case»)==1 && transform.position[0] < thisComp.width/2) {*
100
} else {50}»

Voilà encore un nouvel opérateur, celui là fait partie des opérateurs logiques : et «&&». Il y a aussi le ou «||» (qui se fait avec la touche 6 du clavier) et la négation «!».

Il est donc possible de mettre autant de conditions que l'on veut en combinant les opérateurs logiques. Il est possible d'utiliser des parenthèses pour préciser la priorité :

- Sur un solide, en opacité, mettez l'expression suivante :
«*m=12;*
n=8;
if (m<10 && n==7 || m==12) {100}
else {50}»

Le solide est allumé. En effet, pour être allumé il doit remplir SOIT la condition m<10 et n=7, SOIT m=12. La deuxième des deux conditions est remplie, il est donc allumé. Ajoutons des parenthèses de la façon suivante :
«*m=12;*
n=8;
if (m<10 && (n==7 || m==12)) {100}
else {50}»

Le solide est maintenant éteint, puisqu'il doit remplir deux conditions, la première, m<10, n'est pas remplie, tandis que la deuxième n=7 ou m=12 est remplie.

IV - Fonctions

Nous savons maintenant tout faire avec les expressions ! Il ne manque que les fonctions, pour les plus compliquées des expressions, qui permettent d'accéder à certaines propriétés de After Effects, certains calculs, ou encore de simplifier certaines opérations répétitives...

C'est quoi, une fonction ?

Nous en avons déjà utilisée une... souvenez vous de «*radianToDegrees(valeur)*» qui utilisait la valeur et la convertissait en degrés. C'est ça une fonction, c'est le fait de prendre une entrée ou plusieurs entrées (ici la valeur, mais ça peut être des objets, des tableaux...), de faire un certain nombre d'opérations avec ces entrées, et d'en resortir (retourner on dit en programmation) quelque chose. Ici en l'occurence, la valeur convertie en degrés.

Il y en a de nombreuses toutes faites et prêtes à être utilisées dans After Effects, accessible comme on l'a vu précédemment par la bibliothèque de fonctions. Chacune est définie par son nom, puis entre parenthèses, ses arguments (ce qu'on doit rentrer). (attention, il y a aussi de simples objets dans la bibliothèque, tout ce qui n'a rien entre parenthèse...).

Nous allons en aborder trois très utiles à titre d'exemples.

Wiggle
wiggle(freq, amp, octaves = 1, amp_mult = .5, t = time)

C'est une fonction qui crée une valeur qui varie dans le temps, selon une certaine fréquence et une certaine amplitude. On distingue deux types d'arguments : les obligatoires (freq et amp) et les optionnels (les autres, repérables au signe «=»). C'est à dire que dans l'usage le plus courant de la fonction, écrire par exemple «*wiggle(15,20)*» est suffisant, les autres arguments sont remplacés par ceux par défaut. C'est donc exactement équivalent à «*wiggle(15,20,1,.5,time)*».

Pour tester, essayez sur la position d'un calque. On constate en passant que le type de valeur que crée le wiggle est bien un tableau à deux dimensions, puisque ça fonctionne sur la position. En fait, le wiggle s'adapte à la valeur en cours, et la variation se fait autour de la valeur existante.

freq : c'est la fréquence, en Hertz, autrement dit en nombre de changements par secondes, à laquelle la variation de valeur se fait. La vitesse du wiggle en quelque sorte.

amp : c'est l'amplitude du wiggle, sa force.

octave : cette valeur permet de rendre le wiggle fractal. C'est à dire, plus simplement, de mettre un wiggle sur le wiggle. La valeur par défaut 1, fait une variation correspondant à la fréquence et l'amplitude des deux premiers arguments, la valeur 2 ajoute par dessus une variation a une fréquence deux fois plus élevée et à une amplitude multipliée par le quatrième argument, amp_mult.

amp_mult : cette valeur doit être comprise entre 0 et 1, c'est le multiplicateur de l'amplitude des wiggles supplémentaires définis par l'octave.

t : cette valeur définit comment le wiggle est géré dans le temps. La valeur par défaut, time, permet d'utiliser le déroulement «logique» de la timeline. Si l'on entre une constante, le wiggle ne se fait plus (il faut que cette valeur varie dans le temps). Si l'on met par exemple «*time*2*» ca revient à doubler la fréquence. Si l'on met «*time+2*», ça «décalle» le wiggle de deux secondes dans le temps.

Length
length(point1, point2)

Cette fonction n'est pas très compliquée, mais très utile. Les deux arguments sont des tableaux à deux ou trois dimensions, représentant des coordonnées. La fonction retourne la distance entre les deux points. Elle permet par exemple de mesurer la distance entre une caméra et le point d'ancrage d'un calque, permettant de faire varier l'opacité ou la couleur du calque par rapport à sa distance avec la caméra. Essayons, vous verrez à quel point c'est utile :

- Créez une caméra, en position, mettez la valeur de Z à 0.
- Créez un solide, mettez le en mode 3D, et mettez le en Z à 1000 (ou -1000, suivant dans quel sens pointe la caméra)
- Dans l'opacité du solide, mettez cette expression :
«*distance = length(thisComp.layer(«Camera 1»).transform.position,transform.position);*
if(distance<1000) { distance/10 }
if(distance>1500) { 200-distance/15 }
if(1000<distance && distance<1500) { 100 }»

Déplacez votre solide en Z, ça se passe de commentaires...

toWorld
toWorld(vec, t = time)

Cette fonction a deux aspects très utiles... L'un spatial, qui permet de convertir des coordonnées pour qu'elles soient absolues, l'autre temporel. Un exemple pour comprendre l'utilité de la fonction :

- Créez un solide noir, appliquez-y l'effet lumière parasite.
- Créez un objet nul.
- Avec une expression, liez le centre de la lumière parasite à la position de l'objet nul, puis déplacez cet objet nul.
- Jusque là, tout va bien. C'est même pratique. Mais maintenant, créez un deuxième nul, et liez le premier à ce nouveau nul. Ah mince ! Regardez bien la valeur de la position de l'objet nul : elle change quand vous le liez à un autre objet, donc forcément, le centre de la lumière parasite adopte cette nouvelle valeur ! Mais pourquoi ? Parce que dans After, les coordonnées de position sont données par rapport au parent ! Si il n'y a pas de parent, tout va bien, les coordonnées sont par rapport à la composition. Mais si il y a un parent, la position [0,0] se retrouve sur le point d'ancrage de ce parent... C'est la qu'intervient notre fonction toWorld. toWorld s'utilise de la façon suivante : «*objet.toWorld(coordonnées, t = time)*». Les coordonnées, ce sont les coordonnées à ramener en coordonnées absolues. L'objet, c'est l'objet par rapport auquel les coordonnées avant convertion sont traitées (dans notre cas, le deuxième objet nul, parent du premier). L'usage est donc simple :

- Dans le centre de la lumière parasite mettez l'expression :
«*thisComp.layer(«objet nul 2»).toWorld(thisComp.layer(«objet nul 1»).transform.position)*»

toWorld s'utilise de la façon suivante : «*objet.toWorld(coordonnées, t = time)*». Les coordonnées, ce sont les coordonnées à ramener en coordonnées absolues. L'objet, c'est l'objet par rapport auquel les coordonnées avant convertion sont traitées (dans notre cas, le deuxième objet nul, parent du premier). L'usage est donc simple :

- Dans le centre de la lumière parasite mettez l'expression :
«*thisComp.layer(«objet nul 2»).toWorld(thisComp.layer(«objet nul 1»).transform.position)*»

Maintenant tout fonctionne. Mais il faut avouer que ce n'est pas très pratique... Il faut savoir par rapport à quoi les coordonnées de position sont données, ce n'est pas forcément évident. Par contre, les coordonnées d'un point d'ancrage sont toujours données par rapport à son calque. Donc c'est plus facile de transformer les coordonnées du point d'ancrage en coordonnées absolu, plus besoin de se soucier des liens entre les calques. Utilisons maintenant l'expression :
«*thisComp.layer("objet nul 1").toWorld(thisComp.layer("objet nul 1").transform.anchorPoint)*» et le tour est joué !

Quant au deuxième argument, il permet de récupérer des coordonnées à un moment donné (valeur en secondes). La valeur par défaut «time» représente l'instant courant.

Il y a donc plein de fonctions dans After, mais ne peut-on pas créer les notres ?

La création de fonction est assez simple : il suffit d'utiliser la syntaxe suivante :
«*function moyenne(argument1,argument2) {*
A = argument1 + argument2;
return A/2 ;
}»

Le mot «return» définit ce qui sera retourné par la fonction. Ici, le résultat de l'opération «A/2».

Et l'utilisation de la fonction ainsi créée se fait de la façon suivante :
«*moyenne(10,12)*», qui retournera la valeur 11.

Un exemple de fonction utile, revenons à nos trois solides de la page 10 :
- Créez trois solides (d'une taille plus petite que celle de la compo, pour bien voir leur position)
- Animez la position des deux premiers, et ajoutez l'expression suivante en position sur le troisième :
«*positionA = thisComp.layer(«calque1»).transform.position;*
positionB = thisComp.layer(«calque2»).transform.position;
(positionA + positionB) / 2»

On pourra remplacer cette expression par la suivante, maintenant qu'on connait toWorld :
«*positionA = thisComp.layer("calque1").toWorld(thisComp.layer("calque1").transform.anchorPoint);*
positionB = thisComp.layer("calque2").toWorld(thisComp.layer("calque1").transform.anchorPoint);
(positionA + positionB) / 2»

Ce qui fera fonctionner l'expression même si les calques ont des liens. Et en créant une fonction on pourra encore simplifier cette expression :
«*function positionAbsolue(calque) {*
calque.toWorld(calque.transform.anchorPoint);
}
(positionAbsolue(thisComp.layer("calque1"))+ positionAbsolue(thisComp.layer("calque2"))) / 2»

Ce qui est très utile si on doit manipuler de nombreux calques...

ANNEXE

Liste des opérateurs :

Calcul :

+ addition
- soustraction
/ division
* multiplication
% modulo
= affectation

Affectation :

+= addition
-= soustraction
/= division
*= multiplication
%= modulo

Incrémentation :

++ addition
-- soustraction

Comparaison :

< strictement inférieur
> strictement supérieur
<= inférieur
>= supérieur
== égalité
!= différence

Logiques :

|| ou
&& et
! négation

Classement de ces opérateurs par priorité

priorité maximale

()	[]				
--	++	!			
*	/	%			
+	-				
<	>	<=	>=		
==	!=				
&&	||				
=	+=	-=	*=	/=	%=

,

priorité minimale