

Leçon 19

Les fonctions approfondies de TVPaint Animation

Dans cette leçon, vous allez apprendre à :

- Créer des fenêtres personnalisées.
- Utiliser les plug-ins à votre disposition.
- Utiliser le langage de script « George ».

Les fenêtres personnalisées

Il est possible de créer vos propres panneaux et icônes personnalisées dans TVPaint Animation, dans le but de transformer l'interface selon vos besoins.

Pour avoir accès aux fenêtres personnalisées de TVPaint Animation, cliquez sur le bouton cerclé de la barre d'*Outils* (voir ci-dessous).

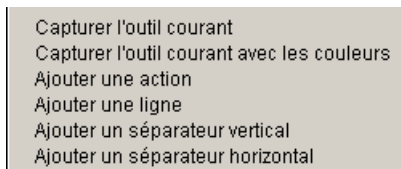


Les fenêtres personnalisées sont semblables à celles représentées ci-contre : de nombreuses icônes sont visibles, chacune ayant une fonction bien spécifique.

Par défaut, quand vous créez une nouvelle fenêtre personnalisée, cette dernière est vide mais nous allons voir comment la remplir un peu plus loin.





Un panneau fraîchement créé se présentera comme dans l'image ci-contre et sera nommé par défaut *Tool Bar*.



Un clic droit à l'intérieur d'une zone vide d'une fenêtre personnalisée ouvre le menu ci-contre.

Vous pouvez :

- * *Capturer l'outil courant* qui vous permettra de stocker votre outil actuel avec ses réglages et connexions: aérographe, plume, brosse personnalisée, etc ... Cette action crée un bouton dans le panneau courant, bouton dont l'icône est celui de l'outil capturé et qui permet d'accéder rapidement à l'outil précédemment capturé. Notez que cette option pour capturer l'outil courant est accessible également en pressant simplement le bouton  du panneau *Tool Bin* présent par défaut.

- * *Capturer l'outil courant avec les couleurs* fait la même chose que l'action précédente à la différence que l'outil est stocké avec les couleurs A et B définies au moment de la capture. De la même façon que précédemment, cette option est accessible au travers du bouton  du panneau *Tool Bin*.

- * *Ajouter une Action* ou succession d'actions.

- * *Ajouter une ligne* d'icônes à remplir.

- * *Ajouter des séparateurs verticaux et/ou horizontaux* entre vos icônes (cliquez avec le bouton droit sur ces derniers pour les retirer ...). Les *séparateurs horizontaux* peuvent être réduit pour gagner de la place à l'écran.

Ajouter une action

Ajouter une action se fait à l'aide du panneau en bas à gauche de cette page :

* A l'aide des deux premiers champs de texte, vous pouvez définir un nom et un commentaire d'aide en ligne pour votre future fonction personnalisée.

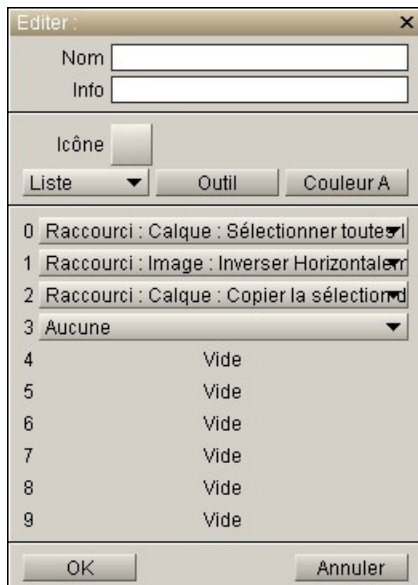
* Les boutons et menus *Liste*, *Outil* et *Couleur A* vous permettront de choisir l'icône correspondante à l'action que vous allez créer.



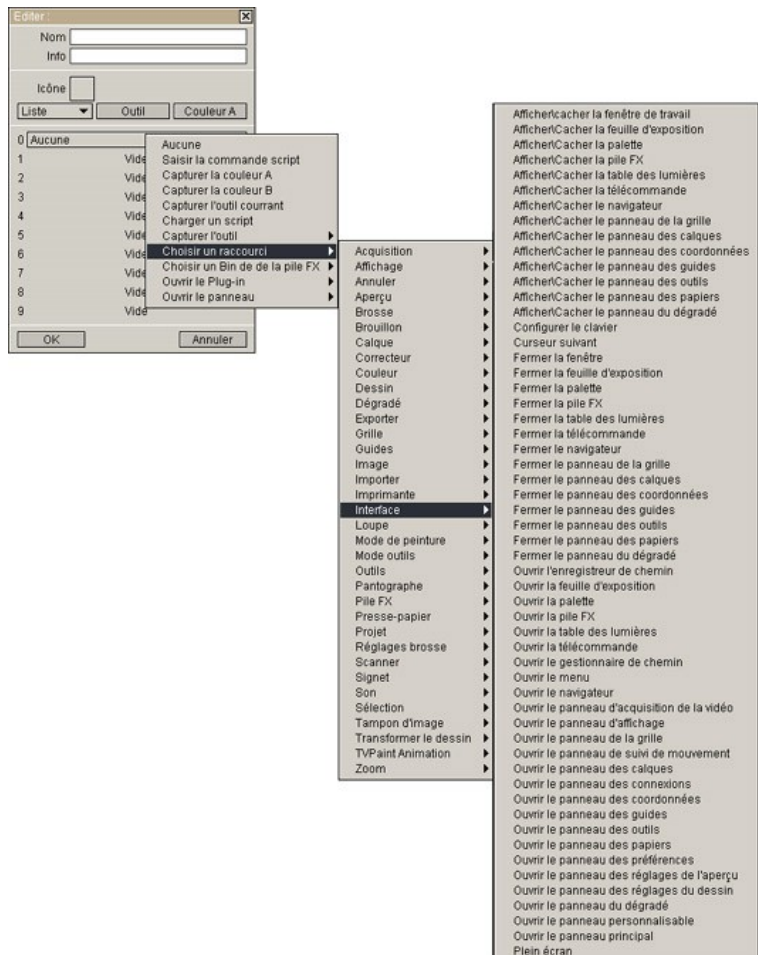
Ci-contre la liste d'icônes fournies avec TVPaint Animation.

Pour créer une nouvelle icône, il vous suffit de la dessiner, puis de la découper pour en faire une brosse et de sélectionner le bouton *Outil*.

* Les lignes numérotées sont les actions qui seront exécutées l'une après l'autre lorsque vous cliquerez sur l'icône de votre fonction.



Ci-dessus un exemple d'action qui sélectionne toutes les images du calque courant, les inverse et copie le tout dans un nouveau calque.



Les différentes actions disponibles

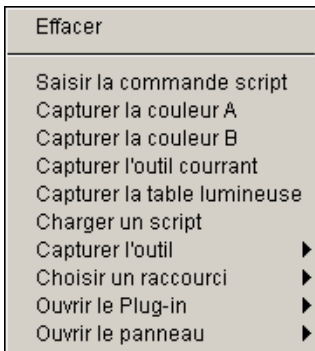
Plusieurs choix s'offrent à vous lorsque vous décidez de créer une action. Vous pouvez :

* Choisir un raccourci vers l'une des nombreuses fonctions du logiciel (voir page précédente).

* Choisir un fichier Bin stocké dans la pile-FX pour le réutiliser.

* L'action *Ouvrir le Plug-in* vous permet de mettre en place un bouton pour lancer rapidement des plug-ins. (ces derniers sont détaillés dans la section suivante).

* L'action *Ouvrir le panneau* vous permet quant à elle de créer des dépendances entre vos panneaux, c'est à dire que vous pouvez afficher des panneaux secondaires depuis un panneau principal par exemple. Cela peut se révéler pratique pour organiser vos brosses personnalisées ou le contenu de vos différents *Bins*.



- * Capturer la couleur A.
- * Capturer la couleur B.
- * Capturer l'outil courant, avec la couleur A, B ou avec ces deux couleurs.
- * Appeler un script d'automatisation George.
- * Saisir une commande script (par exemple : TV_LayerAnim, TV_Circle, TV_Airbrush, etc ...).

Les scripts et commandes George seront abordés un peu plus loin dans cette leçon.

Pour gérer les actions créées.

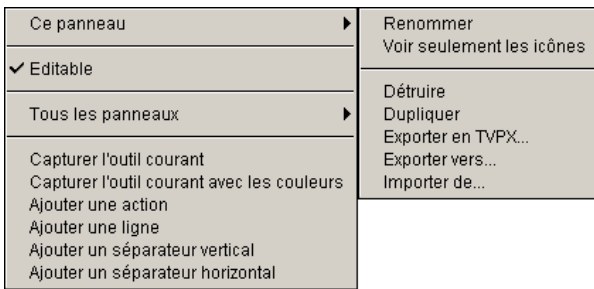
Une fois vos boutons créés dans la fenêtre personnalisée, d'autres options s'offrent à vous :

- * Par simple sélection dans votre fenêtre personnalisée, vous pouvez couper ou copier les icônes/actions que vous venez de créer. Il vous suffit alors d'employer le menu précédemment décrit pour pouvoir « coller » vos icônes/actions ultérieurement.
- * Par un clic droit sur une icône/action de la fenêtre personnalisée, vous pouvez éditer, dupliquer ou supprimer celle-ci.

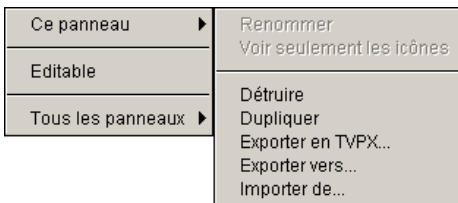


Pour gérer les panneaux

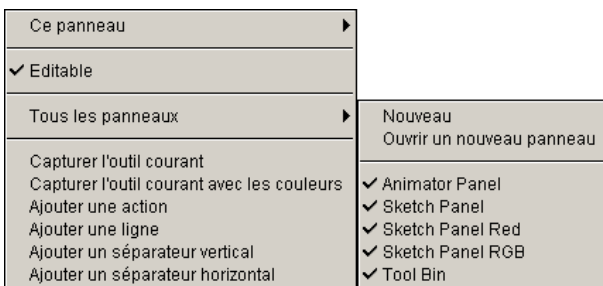
Restent les options propres aux panneaux en eux-mêmes. Dans le sous-menu ci-dessous, accessible par un clic droit dans une zone vide votre fenêtre personnalisée, vous pouvez :



- * *Renommer* la fenêtre personnalisée courante.
- * *Voir seulement les icônes* (sans le texte donc).
- * Disposer votre fenêtre sous la barre d'outils (*Dock*).
- * *Détruire* la fenêtre courante.
- * *Dupliquer* la fenêtre dans son ensemble.
- * Exporter en tant que fichier TVPX.
- * *Importer / Exporter* la fenêtre courante.



Un panneau personnalisé peut être marqué comme *Editable* ou *Non-Editable*.
Les actions non permises dans un panneau non-éditable sont grisées.



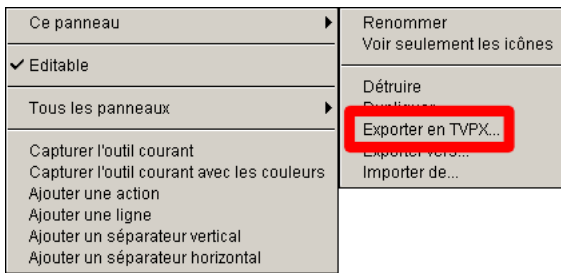
- Dans ce second sous-menu vous pouvez :
- * *Créer une nouvelle fenêtre personnalisée* (vous pouvez les créer en nombre illimité).
 - * *Charger un nouveau panneau* à partir de son fichier .bin.
 - * *Visualiser* le nom des fenêtres personnalisées visibles à l'écran.



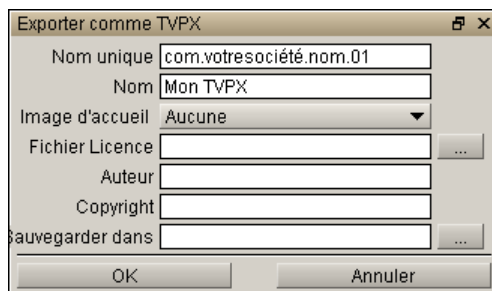
Vous pouvez également définir un raccourci clavier pour chaque action créée par le biais du tableau des raccourcis clavier !

Comment utiliser le format de fichier TVPX.

Il est possible d'exporter les fenêtres personnalisées dans un fichier TVPX. Ce format rend l'échange des fenêtres personnalisées plus aisé. Tout le contenu de la fenêtre personnalisée sera exporté dans un unique fichier TVPX.



Une fois votre fenêtre prête à être exportée, choisissez l'option *Exporter en TVPX...*



Un panneau vous demandera certaines options du fichier exporté.

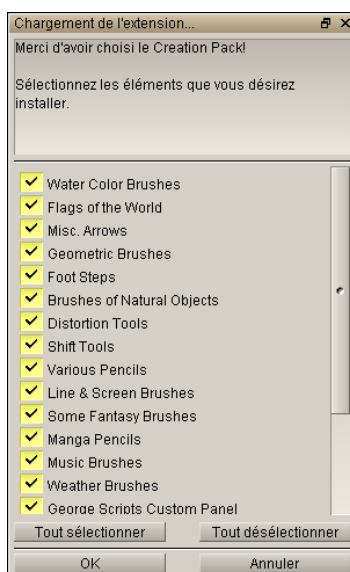
- * Le premier champ est un nom unique utilisé de façon interne par TVPaint Animation. Évitez de donner le même nom à deux fenêtres distinctes exportées.
- * Le champ *Nom* est le nom de la fenêtre tel que présenté à l'utilisateur.
- * L'Image d'accueil sera affichée lors de l'installation du fichier TVPX. Les choix sont:
 - _ *Aucune* : aucune image ne sera affichée.
 - _ *Image courante*: L'image courante du projet sera utilisée.
 - _ *Brosse personnalisée* : L'image courante de la brosse personnalisée sera utilisée.
 - _ *Capturer le panneau* : L'image de la fenêtre personnalisée sera utilisée.
- * Le champ Fichier Licence permet de sélectionner un fichier de texte qui sera affiché en tant que licence que l'utilisateur devra accepter pour pouvoir installer le fichier TVPX.
- * L'auteur et un court copyright.
- * Et enfin, le chemin du fichier TVPX à créer.

Toutes ces options seront utilisées lors de l'installation du fichier TVPX.

Mais comment installer un fichier TVPX ? Il suffit de le glisser-déposer sur la fenêtre de TVPaint Animation !



Une fenêtre apparaît pour présenter le contenu du fichier TVPX (avec l'éventuelle licence, l'auteur, le copyright, une image d'accueil), cliquer sur le bouton *Suivant*> pour passer à la fenêtre suivante, après avoir éventuellement accepté la licence.



Dans cette seconde fenêtre, il est possible de voir tous les contenus du fichier TVPX et de choisir lesquels seront installés. Ensuite, un clic sur *OK* va les installer.

Utiliser les plug-ins PRO

Pour utiliser l'un des plug-ins à votre disposition, vous devez :

- * Ouvrir une fenêtre personnalisée.
- * Créer un nouveau bouton dans une fenêtre personnalisée.
- * Sélectionner le plug-in souhaité comme action à effectuer pour ce bouton.

Le panneau relatif au plug-in sera alors visible à l'écran si vous cliquez sur ce bouton.



Le plug-in *Canon* n'est disponible que si vous l'avez sélectionné lors de l'installation du logiciel. Il n'est disponible que sur PC à ce jour.

Le plug-in *Color Factory*

Le plug-in *Color Factory* permet de modifier à l'aide de formules mathématiques la valeur des composantes *Rouge*, *Vert*, *Bleu* et *Alpha* des pixels des images sélectionnées.



En vertu du principe de la synthèse additive des couleurs, toute couleur est un mélange de rouge, de vert et de bleu (plus ou moins foncés) selon des proportions précises.

Tout pixel de l'écran possède une couleur qui peut être retranscrite numériquement dans un système donné. Par exemple, l'onglet *curseur* du panneau *palette* donne les valeurs correspondant à la couleur A dans les systèmes R,V,B et T,S,L.



Chaque champ textuel du panneau *Color Factory* correspond à l'une de ces composantes (R, V, B et Alpha) et peut accueillir une formule mathématique. Les nouvelles valeurs des composantes seront calculées en fonction des formules inscrites (La composante *Alpha* indique à quel point le pixel est transparent).

Les variables que vous pouvez employer dans vos formules sont les suivantes (toutes en lettres minuscules) :

Les variables :	
r	Valeur de la composante <i>Rouge</i> du pixel
g	Valeur de la composante <i>Verte</i> du pixel
b	Valeur de la composante <i>Bleue</i> du pixel
a	Valeur de la composante <i>Alpha</i> du pixel
x	Coordonnée x du pixel
y	Coordonnée y du pixel
w	Largeur en pixels du projet courant
h	Hauteur en pixels du projet courant

Les opérateurs arithmétiques à votre disposition sont les suivants :

Opérateurs arithmétiques	Exemples :
*	multiplication $18 * 3 = 54$
/	division $18 / 3 = 6$
+	addition $18 + 3 = 21$
-	soustraction $18 - 6 = 12$

Il est possible d'utiliser des parenthèses () pour définir des priorités de calculs. Si vous n'en utilisez pas, les calculs se feront de gauche à droite.

Par exemple : $1+2*3 = 9$ $1+(2*3) = 7$

Voici quelques exemples utilisant le plug-in *Color Factory* :

* Pour commencer, une image normale sans modification :



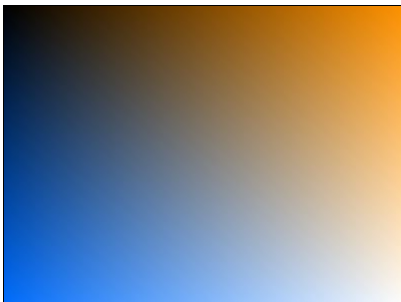
* Ici un négatif de la photographie précédente a été obtenu à l'aide de notre plug-in :



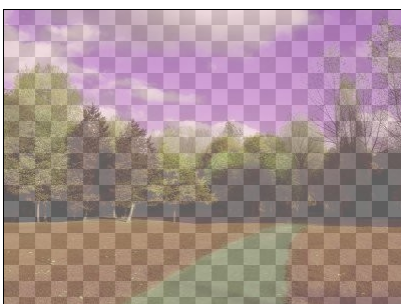
* Ici Les composantes rouges et bleues de notre image ont été interverties : le ciel est devenu rouge-orangé, le chemin est passé du rouge au bleu. La composante verte est inchangée.



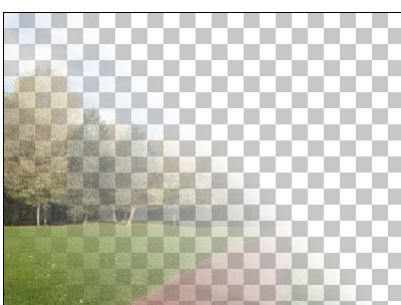
* Le plug-in Color Factory peut également être utilisé pour créer des dégradés de couleur élaborés. Les formules utilisées sont, dans ce cas, plus complexes :



* Ci-dessous, des opérations simples ont été effectuées sur des canaux de couleur rouge et vert intervertis :



* Pour terminer, voici un dégradé en diagonale (du haut à droite vers le bas à gauche) pour la composante alpha (composante de transparence) de notre image :



Le plug-in *Canon* (PC seulement)

Le plug-in *Canon*, permet de récupérer dans un calque TVPaint Animation des photographies prises avec un appareil photo de marque *Canon*.

Il est très pratique pour les personnes désirant travailler en Stop-Motion directement avec le logiciel.

Voici, ci-dessous la marche à suivre pour faire vos premiers essais avec le *Plugin Canon* et TVPaint Animation (pour plus de détails, veuillez consulter notre forum en ligne).

* conditions préalablement requises :

- Le pilote *Canon* de votre appareil numérique doit être installé dans votre système (Windows uniquement à ce jour).
- Vous devez posséder un appareil *Canon* compatible (voir liste en Annexe) alimenté et connecté à l'un des ports USB de votre machine.
- Enfin, il vous faut mettre l'appareil en mode *lecture*. N'utilisez pas le mode *prise de vue*, afin de pouvoir le piloter depuis votre machine.



Lisez la leçon 5 pour en savoir plus sur le panneau d'*acquisition vidéo*



Il est recommandé, une fois la session d'acquisition terminée, de fermer les deux fenêtres d'acquisition vidéo avant de quitter le programme.

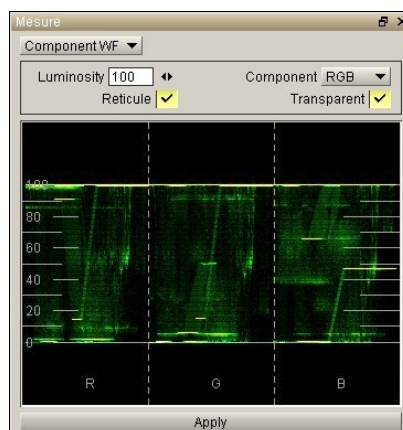
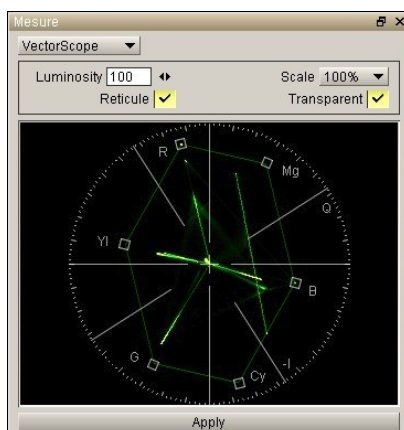
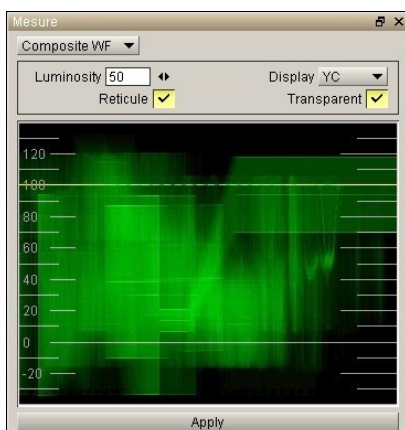
Le plug-in *Direct Show*

DirectShow est une application de programmation d'interface de Microsoft qui contient plusieurs bibliothèques logicielles. C'est un composant de DirectX qui autorise la gestion de fichiers multimédias. Le Plug-in DirectShow fournit un support pour les matériels compatibles USB, SCSI, FireWire, ... (théoriquement tout matériel compatible avec la norme WDM : Windows Driver Mode)

Le plug-in *Waveform*

De manière générale, un oscilloscope est un appareil de mesure permettant de contrôler et mesurer des signaux électriques périodiques. S'il est calibré pour la visualisation de signaux vidéo, on parle de moniteur de forme d'ondes.

Dans le milieu de l'audiovisuel, ces derniers sont utilisés pour s'assurer que le signal vidéo transmis présente des caractéristiques conformes aux normes de télédiffusion en vigueur.



Notre Plug-in Waveform permet aux professionnels de l'audiovisuel de retrouver leurs outils usuels. Grâce à lui, il devient plus simple de gérer calibrations et corrections colorimétriques, de vérifier l'intégrité d'un signal, contrôler la validité des signaux numériques, ...

En fonction de votre choix dans le menu déroulant, il peut-être utilisé en tant que :

- Moniteur de forme d'ondes (encore appelé moniteur de profil).
- Analyseur de composantes.
- VectorScope.



Les différents affichages proposés par ce plug-in sont calculés comme si tous les calques étaient fusionnés.

Voici quelques options spécifiques aux différents affichages :

- * Le VectorScope accepte deux échelles de calibrage : 100% et 75%.
- * L'analyseur de composantes peut fonctionner en mode R,V,B ou Y,U,V.
- * Le moniteur de profil peut afficher le signal de Chrominance, Luminance ou les deux à la fois.

Restent les options communes à tous les modes de visualisation :

- * Il est possible de régler la luminosité de l'affichage à l'aide d'un mini-ascenseur.
- * Le bouton *réticule* gère l'affichage des différents axes et canaux.
- * Le bouton *appliquer* permet de copier sur l'image courante les résultats obtenus dans le panneau *WaveForm*.
- * Si le bouton *transparence* est coché, le fond noir ne sera pas pris en compte si vous appuyez sur le bouton *appliquer*.

Utiliser le langage de script « George »

Introduction : Qu'est-ce que George ?

George est un langage de programmation qui permet d'employer les nombreuses fonctions de TVPaint Animation. Tout programme rédigé dans ce langage est appelé « Script George ». D'autres logiciels emploient parfois le terme de « Macro ».

Un *Script George* permet d'exécuter des instructions propres au logiciel TVPaint Animation dans un ordre précis. Par exemple : créer un calque, dessiner un cercle à l'endroit de votre choix, sauvegarder le travail en cours, ajouter un marqueur, etc ...

Ces *Scripts* sont généralement utilisés afin de gagner du temps lors de l'exécution de tâches répétitives. Les utilisateurs expérimentés utilisent les commandes TVPaint pour la création de plug-ins ".dll".

Des *Scripts George* sont fournis en guise d'exemple avec le logiciel, mais vous pouvez également créer vos propres scripts et en échanger avec d'autres utilisateurs de TVPaint Animation via le forum de TVPaint <http://www.tvpaint.com/forum/index.php>

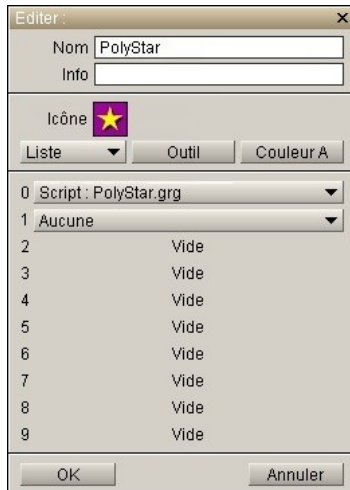
Retrouver, Editer et Utiliser les Scripts George.

Après installation du logiciel, les *Scripts George* sont disponibles dans le répertoire :

C:/program files/TVPaint Animation/George sous Microsoft Windows.
Applications/TVPaint Animation/Contents/Ressource/George/ sous Apple OS-X.

Ces *Scripts George* se présentent sous la forme de fichier texte ASCII et ont une extension de fichier « .grg ». Il est possible de les éditer ou de les créer à partir de n'importe quel éditeur de texte, par exemple *Notepad* sur Windows, *Text Edit* sur Mac OS-X ou *Gedit* sur Linux

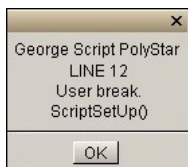
Les scripts sont théoriquement utilisables sur Windows comme sur MAC et Linux et ce quelle que soit la machine où ils ont été créés à l'origine.



Pour utiliser l'un des *Scripts George* à votre disposition, vous devez :

- * Ouvrir une fenêtre personnalisée,
- * Créer un nouveau bouton dans celle-ci,
- * Sélectionner l'action - *choisir un script* - pour ce bouton,
- * Choisir un fichier avec extension « .grg » dans l'explorateur de fichier.

Si vous cliquez sur le bouton ainsi créé, le *Script George* sélectionné sera alors exécuté.



A tout moment, il est possible de stopper l'exécution d'un *Script George* en appuyant sur la touche [Esc] de votre clavier.

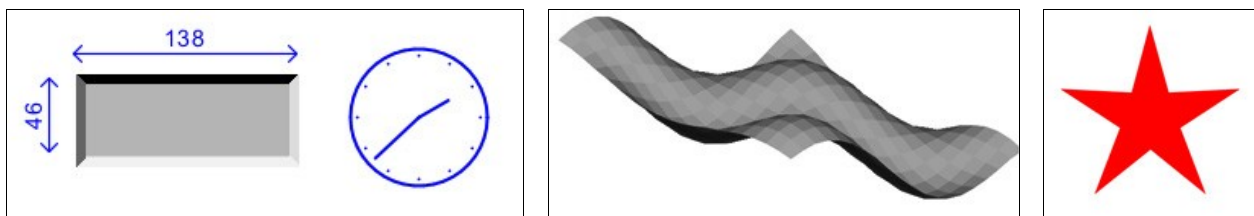
Un panneau s'affiche alors, vous indiquant à quelle ligne du script l'exécution a été stoppée. (voir ci-contre)

Une fois un *Script George* utilisé, l'option *Undo* du panneau principal permet d'annuler la totalité des changements effectués par celui-ci.

Quelques exemples de *Scripts George*

Un *Script George* peut nécessiter une action de la part de l'utilisateur pour commencer à fonctionner. Par exemple :

- * Le *Script George* nommé *long.grg* attend que l'utilisateur trace un segment à l'écran puis renvoie la longueur de ce dernier.
- * Le *Script George* nommé *clock.grg* attend que l'utilisateur trace un cercle à l'écran puis dessine une horloge dont la taille correspond au cercle tracé.
- * Le *Script George* nommé *bouton2.grg* attend le tracé d'un rectangle à l'écran puis dessine un bouton en relief à l'intérieur de celui-ci.
- * D'autres *Scripts George* ne nécessitent aucune action de la part de l'utilisateur. C'est le cas du *Script George* nommé *Zsurface.grg*.

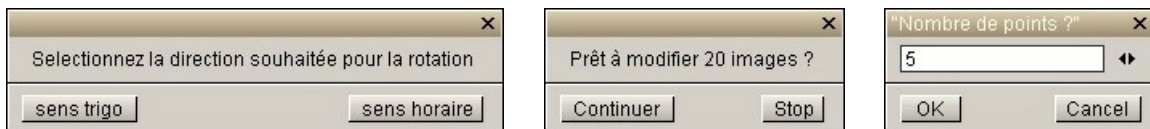


Ci-dessus, le résultat des scripts : *Bouton2*, *Long*, *Clock* et *Zsurface* et *PolyStar*.

Certains *Scripts George* sont plus complexes et il se peut qu'une ou plusieurs requêtes fassent leur apparition.

C'est le cas du *Script George PolyStar.grg* : Lors de son exécution, une boîte de dialogue vous demande de préciser le nombre de sommets de l'étoile qui sera dessinée. Si vous appliquez le

script à un calque d'animation (après avoir sélectionné ces images), d'autres requêtes feront leur apparition. (voir ci-dessous)



Programmer en langage George

George est un langage de script qui permet notamment d'effectuer :

- des boucles (faire la même chose n fois).
- des tests conditionnels (si telle condition est remplie, alors fais ceci, sinon fais cela).
- des calculs ($a+b$, $\sin(a)$,...).

George n'est pas un langage particulièrement rapide, mais il est flexible et très simple d'utilisation. Sur plusieurs points, il est comparable au langage *Pascal* et à d'autres langages de programmation connus.

Cependant, il n'est pas destiné au développement d'applications complexes comme le sont par exemple les langages C et C++. George est en premier lieu et avant tout un langage de script.

Instructions et commandes

George emploie des instructions classiques pour structurer les scripts : événements conditionnels, tests, boucles, etc... Pour une meilleure visibilité, elles seront ici notées en vert.

Par exemple :

Cos, Sin, Tan, Rnd, ...

pour les calculs

If, Else, End, For, Do, Until, While, ...

pour les tests conditionnels

A la différence des instructions classiques, les commandes de TVPaint Animation commencent toutes par le préfixe « **tv_** » et font appel à des fonctions précises du logiciel. Elles seront ici notées en couleur bleue.

Par exemple :

* La commande **tv_LayerCreate** permet de créer un calque.

* La commande **tv_SaveBrushAnim** sauvegarde la brosse animée courante.

* Il est possible de changer les couleurs de l'onglet *bin* du panneau *Palette* à l'aide de la commande **tv_SetPalette**.

* Etc ... (Vous trouverez en annexe la liste complète des commandes TVPaint et des instructions du langage George.)

Toutes les commandes TVPaint nécessitent des arguments qui leur sont propres et renvoient parfois certaines valeurs en réponse.

Par exemple : La commande **tv_LayerCreate** nécessite un nom pour le calque que vous désirez créer. Cette même commande retourne ensuite les informations relatives à votre calque (opacité, nombre d'images, position dans la ligne de temps, ...) afin que vous puissiez les réemployer et/ou les modifier.

Il serait trop long ici de citer chaque commande et d'énumérer les arguments et les variables retournées pour chacune d'entre elles. Ces nombreux paramètres sont détaillés dans le kit de développement TVPaint Animation, encore appelé SDK. Ce dernier est disponible sur simple demande à l'adresse email suivante : tvpaint@tvpaint.fr



Prenez garde de bien différencier les lignes d'instruction propres au langage George (qui peuvent au besoin commencer par un dièse #) et les lignes de commandes destinées à TVPaint Animation (qui commencent par le préfixe « **tv _** »).

Les lignes de commande agissent sur le logiciel en lui-même et bien souvent sur la fenêtre de projet, alors que les lignes d'instruction George structurent le script (conditionnels, calculs, boucles, ...).



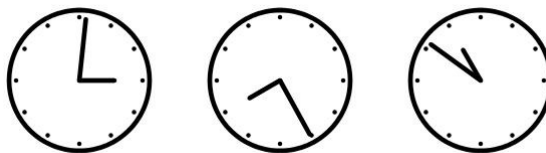
Depuis maintenant près de 10 ans, tous les logiciels issus de la technologie TVPaint possèdent un langage de script.

La compatibilité des scripts entre les différents produits est généralement assurée si vous passez des anciens produits vers les plus récents. (notamment TVPaint Animation)

Par contre, certaines commandes récentes et spécifiques à TVPaint Animation ne sont pas toujours disponibles dans les anciens logiciels utilisant la technologie TVPaint. (Aura, Mirage, ...)

Un exemple de script rédigé avec George.

Le script page suivante provient du fichier *clock.grg*. Il permet de dessiner à l'écran une horloge indiquant l'heure actuelle.



Nous vous invitons à créer un bouton associé à celui-ci dans une fenêtre personnalisée grâce aux indications fournies plus haut. Vous pourrez ensuite essayer facilement ce *Script George* dans la fenêtre de projet.

Passée cette étape, il vous sera possible d'éditer le Script George, de le modifier, de le tester à nouveau, de le rééditer et ainsi de suite. Cela vous aidera petit à petit à apprendre et manipuler le langage George.



Le meilleur moyen pour être plus familier avec les scripts George est probablement d'essayer les nombreux exemples fournis avec le logiciel.

Étudiez-les, modifiez-les très légèrement çà et là. Puis exécutez-les de nouveau. Il sera alors aisé de constater et comprendre l'impact de vos modifications.

Voici le contenu du script `clock.grg` :

```

Param Circle
    // TVPaint Animation attend de recevoir les coordonnées d'un cercle
    // L'utilisateur doit donc dessiner un cercle à l'écran pour que le script
    // soit exécuté.

Parse result command x y r button
    // On répartit les coordonnées du cercle de l'utilisateur dans plusieurs
    // variables.

if cmp(command, "Circle")==0
    tv_Warn "Ce script nécessite le tracé d'un cercle !"
    exit
    // Si aucune coordonnée n'est fournie par l'utilisateur, le script envoie
    // un message d'erreur et s'interrompt.

End

t=time
parse t h m s
    // on répartit les heures (h), les minutes (m), les secondes (s) dans plusieurs
    // variables.
m=m+s/60
    // on ajoute les secondes comme fractions de minutes.
tv_Pen r/30          // Sélectionne un outil à une taille proportionnelle à celle de l'horloge.
tv_Circle x y r      // Dessine le contour de l'horloge.
ang=360/12
d=r-(r/10)
#for i=0 to 360-ang step ang
    // Une boucle permet de dessiner les 12 repères de l'horloge.
    #a=cos(i)*d
    #b=sin(i)*d
    tv_Dot x+a y-b 0
#end

i=h*360/12          // Dessine l'aiguille des heures.
d=r/2
a=sin(i)*d
b=cos(i)*d
tv_Line x y x+a y-b

i=m*360/60         // Dessine l'aiguille des minutes.
d=r-(r/8)
a=sin(i)*d
b=cos(i)*d
tv_Line x y x+a y-b

    // Fin de l'exemple.

```

La description des commandes

Voici quelques informations au sujet des commandes du script page précédente :

Ci-dessous, chaque commande sera suivie par la syntaxe des variables qu'elle requiert. Par exemple, pour dessiner un segment d'un point à un autre, vous devez préciser les coordonnées **(x1,y1)** et **(x2, y2)** des points situés aux extrémités du segment.

La syntaxe pour dessiner un tel segment sera donc :

tv_Line **x1, y1, x2, y2** [0/1]

Le paramètre [0/1] indiquant que vous pouvez utiliser au choix le bouton gauche [0] ou le bouton droit [1] de votre souris.

Ce paramètre est optionnel. Par défaut, le bouton gauche est celui qui est choisi.

Pour résumer :

[] les variables entre crochets sont optionnelles.

/ le symbole slash « / » sépare plusieurs options parmi lesquelles une seule peut être employée. Par exemple, 0/1 indique 0 ou (exclusif) 1

tv_Circle **x y r** [0/1]

Cette commande permet de dessiner le cercle de rayon r et de centre le pixel de coordonnées (x,y).

Le paramètre [0/1] indique que vous pouvez utiliser au choix le bouton gauche [0] ou le bouton droit [1] de votre souris.

Le cercle est bien entendu dessiné sur l'image courante du calque courant.

tv_Dot **x y** [0/1]

Cette commande permet d'appliquer l'outil courant (aérographe, plume, brosse, ...) sur le pixel de coordonnées (x,y), dans l'image courante et le calque courant.

Le paramètre [0/1] indique que vous pouvez utiliser au choix le bouton gauche [0] ou le bouton droit [1] de votre souris.

tv_Warn **texte**

Permet d'afficher une fenêtre contenant une case à cocher «ok» et la chaîne de caractères **texte**. Cette dernière doit être entourée par des crochets.

tv_Pen **taille**

Permet de sélectionner l'outil plume avec ses paramètres actuels (puissance, opacité, mode, ...) et de lui imposer la taille voulue.



Notez bien qu'un Script George utilisera TVPaint Animation « tel qu'il le trouvera », c'est-à-dire avec les réglages de paramètres et d'options actuels.

Si, par exemple, vous désirez dessiner un segment rouge à l'écran, il vous faut indiquer plus que les coordonnées de celui-ci dans votre script.

A moins que vous n'ayez explicitement indiqué d'utiliser une couleur rouge, George utilisera la couleur actuelle, quelle qu'elle soit.

De même, si vous ne précisez pas de mode de dessin et celui qui est actuellement sélectionné n'est pas le mode « couleurs », vous pourriez ne pas obtenir la ligne souhaitée.

Quelques autres notions utiles

A la lecture du script précédent, il convient de préciser les points suivants :

1°) Un dièse # placé sur une ligne donnée ne change pas la lecture d'une ligne de programme.

Par exemple, une ligne du type :

```
Pause 20
```

permet une pause de 20 secondes lors de l'exécution du script.

Il en va de même pour la ligne :

```
#Pause 20
```

2°) Vous ne pouvez utiliser qu'une seule instruction ou commande par ligne. Par exemple, une ligne du type :

```
Print "attendons un peu" Pause 20
```

provoquera l'apparition d'un message d'erreur lors de l'exécution du script. Alors qu'une ligne du type :

```
Print "attendons un peu"  
Pause 20
```

ne provoquera pas d'erreur lors de l'exécution du script.

3°) Deux « slashes » // placés sur une ligne invalideront le texte écrit à leur droite. Ceci est très pratique pour placer des commentaires le long de votre script. Par exemple, une ligne du type :

```
Pause 20 // on attend 20 secondes
```

permet une pause de 20 secondes lors de l'exécution du script. Par contre, une ligne du type :

```
// Pause 20
```

ne générera pas de pause lors de l'exécution du script.



TVPaint Animation ne fait pas de différence entre majuscules et minuscules au niveau des commandes et instructions.

Ainsi : **TV_layercreate**, **tV_LaYeRcReAtE**, **tv_LAYERCREATE** représentent la même commande.

L'emploi des variables

Il n'est pas nécessaire de déclarer vos futures variables lorsque vous écrivez un *Script George*. En d'autres termes, les lignes :

```
MonChiffre=10  
Reponse=false
```

ne nécessitent pas de déclarer la variable *MonChiffre* comme étant de type numérique ou la variable *Reponse* comme étant booléenne.



Les variables employées dans les Scripts George ne sont pas sensibles à la casse. Ainsi George interprétera *MonChiffre*, *MONCHIFFRE*, *mOnChlffrE* de la même façon.



Certains noms de variable sont réservés au langage de programmation : Vous pouvez les lire et les utiliser à tout instant, mais il n'est pas conseillé de les modifier.

C'est par exemple le cas des variables *Time* et *Result* qui contiennent respectivement l'heure actuelle et les résultats de contrôle de nombreuses commandes du langage.

Les différents opérateurs à disposition :

Opérateurs Arithmétiques	Exemples :
- négatif	18 -> -18
** exposant	18**3=18*18*18=5832
* multiplication	18*3=54
/ division	18/3=6
+ addition	18+3=21
- soustraction	18-6=12
Opérateurs Logiques	Exemples :
&& et	a&&b
ou exclusif	a b
Opérateurs Apparentés	Exemples :
== égal à	x==6 x==x+2
!= différent de	x!=5 x!=y-3
> plus grand que	x>4 x>y-z
>= plus grand ou égal à	x>=3 x>=z/y
< plus petit que	x<8 x<-y
<= plus petit ou égal à	x<=7 x<=x+5

* L'opérateur « = » est un opérateur d'assignation et permet d'assigner une valeur donnée à la variable de votre choix. Par exemple :

```
MonNombreDeCalque=5
```

* L'opérateur « == » est un opérateur de comparaison entre deux variables, employé lors des tests. Par exemple :

```
if a==5
```

* Pour savoir si deux variables ont la même valeur, l'opérateur « == » ne convient pas. Il vous faut utiliser l'instruction **cmp**. Par exemple, ci-dessous :

```
if cmp(command,"Circle")==0
```

Les Instructions **Parse** et **Param**, les modes de lancement

L'instruction **Parse** permet de scinder une chaîne de caractères en chaînes de caractères plus petites. Par exemple :

```
ArcEnCiel="rouge orange jaune vert bleu indigo violet"
Parse ArcEnCiel couleur1 couleur2 couleur3
tv_warn couleur1            // affiche « rouge »
tv_warn couleur2            // affiche « orange »
tv_warn couleur3            // affiche « jaune vert bleu indigo violet »
```

En général, tout *Script George* débute par l'instruction **Param**. Celle-ci permet de choisir l'action que l'utilisateur va devoir accomplir pour déclencher l'exécution du script : tracé d'un cercle, d'un segment, d'un rectangle.

* La ligne de script

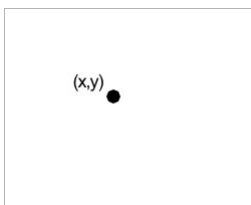
Param None

permettra d'exécuter un Script George sans action initiée par l'utilisateur.

* Les lignes de script

Param Single
Parse result command x y bouton

permettent d'exécuter le script après avoir cliqué sur un point dans la fenêtre du projet en cours.



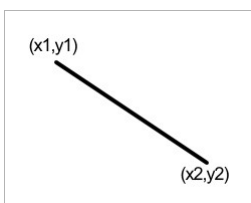
Il sera assigné :

- à la variable *command* la chaîne de caractères « *Single* » si un point est choisi à l'écran par l'utilisateur.
- aux variables *x* et *y* les coordonnées du point.
- à la variable *bouton* : le bouton de la souris employé par l'utilisateur pour placer le point. (1 pour le bouton droit, 0 pour le bouton gauche)

* Les lignes de script

Param Line
Parse result command x1 y1 x2 y2 bouton

permettent d'exécuter le script après le tracé d'un segment sur la fenêtre du projet en cours.



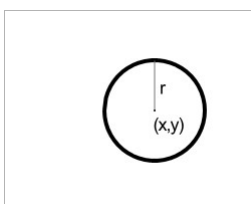
Il sera assigné :

- à la variable *command* la chaîne de caractères « *Line* » si un segment est tracé à l'écran
- aux variables *x1* et *x2* les abscisses des extrémités du segment
- aux variables *y1* et *y2* les ordonnées des extrémités du segment
- à la variable *bouton* : le bouton de la souris choisi par l'utilisateur pour tracer le segment (1 pour le bouton droit, 0 pour le bouton gauche)

* Les lignes de script :

Param Circle
Parse result command x y r bouton

permettent d'exécuter le script après le tracé d'un cercle sur la fenêtre du projet en cours.



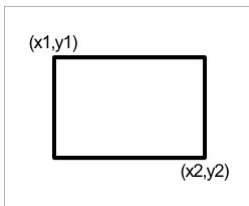
Il sera assigné

- à la variable *command* la chaîne de caractères « *Circle* » si un cercle est tracé à l'écran
- aux variables *x* et *y* : les coordonnées du centre du cercle
- à la variable *r* : le rayon du cercle
- à la variable *bouton* : le bouton de la souris choisi par l'utilisateur pour tracer le cercle. (1 pour le bouton droit, 0 pour le bouton gauche)

* Les lignes de script

```
Param Circle  
Parse result command x1 y1 x2 y2 bouton
```

permettent d'exécuter le script après le tracé d'un rectangle sur la fenêtre du projet en cours.



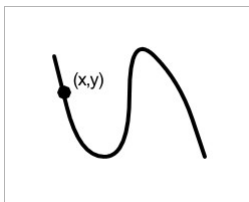
Il sera assigné

- à la variable *command* la chaîne de caractères « *Rectangle* » si un rectangle est tracé à l'écran
- aux variables *x1* et *x2* les abscisses des extrémités du rectangle
- aux variables *y1* et *y2* les ordonnées des extrémités du rectangle
- à la variable *bouton* : le bouton de la souris choisi par l'utilisateur pour tracer le rectangle. (1 pour le bouton droit, 0 pour le bouton gauche)

* Les lignes de script

```
Param Freehand  
Parse result command x y bouton
```

permettent d'exécuter le script après un tracé quelconque sur la fenêtre du projet en cours.



Il sera assigné

- à la variable *command* la chaîne de caractères « *FreeHand* » si un tracé en main libre est effectué à l'écran.
- à la variable *bouton* : le bouton de la souris choisi par l'utilisateur : (1 pour le bouton droit, 0 pour le bouton gauche)

Les Chaînes de Caractères

Il est important de savoir manipuler les chaînes de caractères car elles interviennent dans de nombreuses situations.

Par exemple, dans les lignes ci-dessous, une boucle permet de charger les projets

D:\dessinanime3.tvpp, D:\dessinanime4.tvpp, D:\dessinanime15.tvpp, D:\dessinanime16.tvpp.

```
For nombre=3 to 16  
    NomDeFichier="D:\dessinanime"nombre'.tvpp'  
    tv_loadproject NomDeFichier  
End
```

D'une manière générale, et c'est le cas ci-dessus, il faut utiliser des guillemets simples ('...') ou doubles ("...") pour différencier les variables des chaînes de caractères.

Dans l'exemple ci-dessus :

nombre

est une variable numérique

D:\dessinanime et **.TVPaint**

sont des chaînes de caractères

La variable **NomDeFichier** reçoit la concaténation de **D:\dessinanime** **nombre** et **.tvpp**

Dit autrement : les chaînes de caractères **D:\dessinanime** et **.TVPaint** ainsi que la variable **nombre** ont été mis bout à bout pour former la variable **NomDeFichier**



Il est possible d'utiliser l'instruction **Concat** pour mettre bout à bout chaînes de caractères et/ou variables. Les lignes suivantes ont le même effet :

```
NomDeFichier=nombre'.jpg'  
NomDeFichier=Concat(nombre, '.jpg')
```

Les utilisateurs soucieux de pouvoir employer guillemets simples ou doubles dans leurs chaînes de caractères doivent les entourer des autres guillemets à disposition (voir exemples ci-dessous) :

```
Print ' Mon programme préféré est " TVPaint Animation ' '
           // renvoie des guillemets doubles autour du nom du logiciel.

Print " Mon programme préféré est ' TVPaint Animation ' "
           // renvoie des guillemets simples autour du nom du logiciel.
```

Les tableaux

Comme dans bien d'autres langages de programmation, George rend possible la création de tableaux à une ou plusieurs dimensions.

Le fait d'utiliser des parenthèses ou des crochets accolés à un nom de variable suffisent pour en faire un tableau. L'usage des parenthèses ou des crochets n'a pas d'incidence.

Par exemple :

MonTableau(i) ou MonTableau[i] correspondent au même tableau à une dimension.

MonTableau(i,j) ou MonTableau[i,j] correspondent au même tableau à deux dimensions.

MonTableau(i,j,k) ou MonTableau[i,j,k] correspondent au même tableau à trois dimensions.

Et ainsi de suite ...



Dans le langage George, il n'est pas nécessaire d'indiquer la dimension ou le nombre d'éléments d'un tableau avant de commencer à l'utiliser.

Dans l'exemple ci-dessous, nous avons créé un tableau nommé *couleur*, à une seule dimension, contenant les quatre éléments : "Magenta", "Cyan", "Jaune" et "Noir".

Ensuite nous avons affiché son contenu à l'aide d'une boucle.

```
couleur[1]="Magenta"
couleur[2]="Cyan"
couleur[3]="Jaune"
couleur[4]="Noir"

For i=1 To 4
    tv_warn couleur[i]           // affichera "Magenta", "Cyan", "Jaune", "Noir",
                                // chacun dans un panneau.
End
```

Les fonctions et procédures

Admettons que vous écriviez un script qui nécessite à plusieurs reprises l'emploi des mêmes tests conditionnels, des mêmes instructions, etc ... (Par exemple, ce peut être le cas d'un script qui teste la couleur de plusieurs pixels à l'écran et qui agit différemment en fonction des couleurs obtenues.)

Recopier plusieurs fois les mêmes lignes dans un Script George peut s'avérer vite fastidieux et surtout, cela ne facilite ni la relecture, ni la correction de ce dernier.

Voilà pourquoi il est possible de créer fonctions et procédures :

* Une fonction est un ensemble d'instructions et de commandes qui permettent d'effectuer une tâche précise au sein d'un Script George. Les fonctions sont indépendantes du reste de votre Script George et placées à la fin de ce dernier.

La portion de script distinct de votre ou de vos fonctions sera appelée ici “partie principale du script”. Les fonctions peuvent être utilisées à tout instant par la partie principale du script. Une fonction peut recevoir ou non des paramètres et renvoie toujours un résultat.

* Une procédure est tout simplement une fonction qui ne renvoie pas de résultat.

Voici un exemple de script simple utilisant une fonction:

(Bien entendu, l'emploi de fonctions est plus utile lors de l'écriture de scripts plus complexes et plus longs.)

```
Param Single
Parse result commande abscisse ordonnee bouton
tv_warn TestCouleurNoire(abscisse,ordonnee)

// Partie principale du script ▲
// -----
// Fonction ▼

Function TestCouleurNoire(x,y)
  Local r v b alpha somme
  tv_getpixel x y
  Parse result r v b alpha
  If alpha<>0
    somme=r+v+b
    If somme==0
      Return " la couleur du pixel est noire "
    Else
      Return " la couleur du pixel n'est pas noire "
    End
  Else
    Return " le pixel est transparent "
  End
End
```



La ligne :

```
tv_warn TestCouleurNoire(abscisse,ordonnee)
```

Peut-être remplacée par les deux lignes ci-dessous :

```
Reponse=TestCouleurNoire(abscisse,ordonnee)
```

```
tv_warn Réponse
```

Voici la syntaxe des instructions relatives aux fonctions et procédures :

Function NomDeLaFonction(Variable1, Variable2, Variable3, etc ...)

Il s'agit de l'instruction qui déclare l'existence d'une fonction ou d'une procédure dans un Script George. (Il n'existe pas d'instruction nommée « procédure »)

Cette instruction est suivie du nom donné à la procédure ou à la fonction, puis par le nom des variables de la partie principale du script qu'elle nécessitera.



Les fonctions et procédures doivent toujours être écrites sous la partie principale de votre script.

Appeler une fonction ou une procédure dans la partie principale de votre script s'effectue en indiquant son nom et les variables qui y sont associées entre parenthèses.

Local VariableLocale1, VariableLocale2, VariableLocale3, etc ...

Cette permet de spécifier les variables qui seront utilisées uniquement dans le cadre de votre fonction ou procédure.

Return Resultat

Cette instruction permet de préciser le résultat de votre fonction et de rendre celui-ci disponible pour la partie principale de votre script. Ce peut être une chaîne de caractères, une valeur numérique, un booléen, ... Il est donc possible de le stocker dans une variable, de le réutiliser, ... Cette option est facultative. Si elle n'est pas employée, vous avez créé une procédure.

End

Toute fonction ou procédure se termine par l'instruction « end ».



Les fonctions et procédures du langage George sont récursives, c'est à dire qu'une fonction peut faire appel à elle-même. Cela permet de créer des scripts plus complexes.

A titre d'exemple, dessiner une courbe fractale peut s'effectuer à l'aide d'une fonction récursive. (voir ci-dessous)



Créer des bibliothèques de fonctions et de procédures en langage George

Nous avons vu dans le chapitre précédent qu'il était possible de créer vos propres fonctions et procédures en langage George.

Cela va plus loin puisqu'il est également possible de stocker vos fonctions et procédures dans des fichiers «.grg» et de les réemployer dans d'autres scripts rédigés avec George.

En d'autres termes, vous pouvez créer vos propres bibliothèques.

Pour insérer une de vos bibliothèques dans un Script George, il vous faut écrire la ligne suivante dans votre script :

```
#Include "MaLibrairie.grg"
```

L'instruction **#Include** ira chercher le fichier nommé *MaLibrairie.grg* dans le répertoire contenant le Script George actuel ou dans le répertoire principal de TVPaint Animation.

La ligne suivante permet d'indiquer un chemin d'accès plus précis, toujours à partir des répertoires de base.

```
#Include "MonDossierDeLibrairies/MaLibrairie.grg"
```



Veuillez consulter la section **fichier d'initialisation** de l'Annexe si vous désirez changer le répertoire principal de TVPaint Animation.

Deux exemples de bibliothèques de fonctions relatives aux chaînes de caractères sont disponibles dans le répertoire :

C:/program files/TVPaint Animation/George/Include Applications/TVPaint Animation/George/Include

sous *Microsoft Windows*.
sous *Apple OS-X*.

Elles se nomment *Basic.grg* et *Advanced.grg*

Pour les rendre disponibles dans les scripts George fournis avec le logiciel, il suffit d'ajouter les lignes suivantes en fin de script :

```
# Include "include/basic.grg"  
# Include "include/advanced.grg"
```

Le fichier *Startup.grg*

Au démarrage de TVPaint Animation, (juste après l'affichage des panneaux, mais juste avant de donner le contrôle à l'utilisateur), le Script George *Startup.grg* est lancé. (Il se trouve dans le même répertoire que les autres scripts)

En son absence, rien ne se produit. Dans le cas contraire, vous avez à disposition un fichier de démarrage qui sera exécuté à chaque lancement du logiciel.

Si par exemple, vous renommez le fichier *Logo.grg* en *Startup.grg* le logo TVPaint Développement apparaîtra à chaque mise en route du logiciel.

Vous pouvez également, à l'aide d'un simple Script George, modifier tous les paramètres à votre guise. Par exemple : Charger un projet sur votre disque dur, changer les couleurs courantes, ajouter des calques à votre projet etc...

La création de Plug-in pour TVPaint Animation

Comme expliqué précédemment, George n'est pas le seul langage capable d'envoyer des commandes à TVPaint Animation.

N'importe quel programme peut théoriquement en faire autant. Des exemples utilisant des scripts Excel, des programmes en C ou autres langages sont tout à fait envisageables.

Il est d'ailleurs possible pour les utilisateurs chevronnés ou des tierces compagnies d'écrire des Plug-ins pouvant être utilisés avec TVPaint Animation.

Ces derniers permettent d'ajouter des fonctions personnalisées au logiciel. La création de nouveaux modes de dessin, nouveaux effets ou outils est parfaitement possible.

Les Plug-ins peuvent prendre la forme de fichiers « .dll » (*Dynamic Link Library*), créés à l'aide d'un compilateur. Les fichiers « .dll » ne sont pas lisibles dans un éditeur de texte, contrairement aux fichiers scripts.

Pour plus de détails sur ce sujet : consultez le forum de TVPaint Développement ou demandez le Kit de Développement (SDK) de TVPaint Animation à l'adresse suivante : tvpaint@tvpaint.fr

Ce kit contient le code source des fonctions relatives à l'interface, une documentation et des plug-ins détaillés en exemples.

L'ajout d'arguments

Il est possible d'ajouter des arguments lors du démarrage de TVPaint Animation. Vous pouvez au choix employer :

- des images (indiquez juste le nom et le chemin du fichier),
- des scripts (syntax: script=CheminEtNom.grg),
- des commandes George (syntaxe: cmd=GeorgeCommande).

Par exemple :

```
TVPaint Animation.exe "MonImage" "script=MonScriptCheminEtNom.grg"  
"cmd=MaCmdeGeorge"
```