RagTime et AppleScript – Brève introduction

Jürgen Schell
RagTime GmbH

Traduction française
Nils Frisch

Contenu

À propos de ce document

À l'origine, j'avais écrit cette introduction au pilotage de RagTime avec AppleScript pour la rubrique "Tech Note" de la version allemande du site web de RagTime GmbH. (NdT: Elle a été traduite en langue française par Nils Frisch (Nils.Frisch@wanadoo.fr/NilsFrisch@aol.com) pour favoriser sa diffusion au sein de la communauté des utilisateurs francophones de RagTime.)

À ma connaissance, c'est la meilleure introduction sur ce sujet: en effet, c'est la seule. ;-)

Il était logique que ce document soit aussi en langue anglaise (NdT: la langue d'origine était l'allemand). J'ai donc tenté de le traduire, mais je ne suis pas un natif de cette langue. Ce document n'est pas un modèle de traduction en anglais, mais j'espère qu'il sera profitable à ceux qui veulent piloter RagTime via AppleScript.

Le document n'a pas pour but d'expliquer toutes les possibilités de pilotage de RagTime. Même après avoir lu ce document, il sera nécessaire de regarder le dictionnaire AppleScript de RagTime avec un éditeur de scripts ou de faire quelques enregistrements AppleScript pour en comprendre la syntaxe. En fait, j'essaie de donner un aperçu sur la manière de manipuler via AppleScript les différents types de composants gérés par RagTime. J'essaie de fournir des détails là où j'estime qu'il est difficile de trouver une solution. Dans la mesure du possible, les scripts de ce document évitent de faire appel à l'interface utilisateur. AppleScript n'est pas vraiment fait pour cela. Dans la plupart des cas, les scripts générés à partir du système d'enregistrement AppleScript (bouton "Mémoriser") ne sont valables que pour des situations bien particulières. J'essaie de favoriser les stratégies qui évitent cela.

Les scripts qui utilisent une syntaxe colorée dans ce document ont été copiés et collés à partir de "Script Debugger" de Late Night Software. Cela signifie que je les ai essayés avant des les présenter dans ce document (NdT: quand cela était nécessaire, des parties d'un script ont été traduites en français).

RagTime et AppleScript

Sous Mac OS, RagTime peut être contrôlé par AppleScript de plusieurs manières. Il est possible soit d'automatiser des tâches, soit d'associer RagTime avec d'autres applications.

AppleScript est le langage de script interne de RagTime pour Mac OS. Les scripts peuvent être exécutés en dehors du programme (par exemple dans l'application "Éditeur de Scripts" d'AppleScript) ou exécutés dans RagTime. Dans les deux cas, ils peuvent aussi contrôler d'autres applications.

Pour développer un script, suivez les étapes suivantes.

Premièrement, vous développez dans un éditeur de scripts externe (comme celui d'AppleScript ou d'une marque tierce comme "Script Debugger" de Late Night Software).

Quand votre script est prêt et débogué, copiez entièrement le texte du script édité dans l'éditeur de scripts. Dans RagTime, ouvrez le menu "Extras -> Commandes et Scripts...". Créez un nouveau script et collez le texte du script. Cliquez sur le bouton "Conserver".

Inversement, vous pouvez copier un script de l'éditeur de scripts intégré dans RagTime vers un éditeur de scripts externe pour le travailler.

La plupart des exemples de scripts de ce document sont complets. N'hésitez pas à les copier-coller dans votre éditeur de scripts pour les essayer.

Pourquoi intégrer des scripts dans RagTime?

Alors que le comportement d'un script est le même qu'il soit dans un éditeur de scripts externe ou dans

RagTime, pourquoi devriez-vous les copier dans RagTime?

a) Les scripts dans RagTime sont directement accessibles via le menu "Extras". Aussi, le script peut être attaché dans RagTime à des composants du type Bouton, par exemple.

b) Les parties du script qui contrôlent RagTime s'exécuteront plus rapidement. Les morceaux de script qui contrôlent d'autres applications, comme FileMaker Pro, ne seront pas plus lents. Au final, cela engendre un gain de vitesse.

c) Les scripts peuvent être sauvegardés dans un document RagTime. En contenant tous les éléments indispensables à son bon fonctionnement, un tel document peut être considéré comme une solution sophistiquée à un problème spécifique.

La fenêtre "Commandes et scripts" de RagTime

La fenêtre issue du menu "Extras -> Commandes et scripts..." affiche une liste sur le côté gauche. Les trois principales entrées de la liste sont: "Commandes intégrées", "Accessoires RagTime 5.5" et le titre du document que vous étiez en train d'éditer. (Le numéro dans "Accessoires" peut varier en fonction de la version de votre logiciel RagTime).

Les scripts AppleScript peuvent être sauvés dans l'entrée "Accessoires" ou dans l'entrée correspondant au document courant. Dans le premier cas, les scripts seront disponibles à tout moment dans RagTime. Dans le second cas, les scripts appartiennent au document et seront disponibles seulement si le document est ouvert et au premier plan.

Déplacer un script dans la liste permet de le copier d'une entrée principale à une autre. Double-cliquer sur un script ouvre l'éditeur de scripts de RagTime. Cliquer sur le bouton "Créer" génère un nouveau script vide.

Comme dans l'éditeur de scripts d'AppleScript, la fenêtre de script possède un champ pour saisir un titre et un autre pour les commentaires.

Le commentaire sera affiché dans les bulles d'aide de chaque commande du sous-menu "Extras -> Scripts".

Notez que le bouton "Conserver" n'a pas été intitulé "Sauver": cela n'est pas le fruit du hasard. Si vous cliquez sur le bouton "Conserver", le script sera stocké en mémoire vive avec votre document. C'est seulement lorsque vous sauvegarderez le document lui-même, que le script sera écrit sur le disque (les scripts des accessoires sont sauvegardés sur le disque lorsque vous quittez RagTime).

Le dictionnaire AppleScript de Ragtime

En utilisant n'importe quel éditeur de scripts, vous pouvez ouvrir le dictionnaire AppleScript de RagTime. Il liste tous les objets et toutes les commandes que RagTime supporte. La plus simple façon d'ouvrir le dictionnaire est de glisser-déposer, dans le Finder, l'icône de l'application RagTime sur celle de l'éditeur de scripts. Vous verrez apparaître une fenêtre listant tous les termes AppleScript de RagTime.

La liste est organisée en "suites" dans lesquelles les commandes et les objets sont classés. Certains termes sont en italique. Ce sont les objets - ou les noms au sens AppleScript. En caractère standard, vous pouvez discerner les commandes - ou verbes - qui s'utilisent avec les objets.

Piloter RagTime

Les objets de l'application

Les objets les plus importants de l'application sont les documents ouverts et les fenêtres.

Pour se référer à un document, vous pouvez utiliser son nom:

```
tell application "RagTime 5"
    tell document "Rapport 1" -- en utilisant son nom
        -- ici vient la suite de votre code
    end tell
end tell
```

Cela sollicitera le document intitulé "Rapport 1", qu'il soit au premier plan (document actif) ou en arrière plan. Il suffit tout simplement qu'il soit ouvert.

Pour se référer à un document que vous êtes en train d'éditer, utilisez les numéros d'index:

```
tell application "RagTime 5"
    tell document 1 -- en utilisant l'index
        -- ici vient la suite de votre code
    end tell
end tell
```

"document 1" est toujours le document qui a une de ses fenêtres au premier plan. "document 2" est le document qui a une fenêtre au second (ou plus) plan, etc.

Les fenêtres

Les fenêtres peuvent être sollicitées de la même façon. Notez qu'il peut y avoir plusieurs fenêtres ouvertes pour un même document.

Exemples:

--> Rapport 1, Maquette de feuillet 1
--> Rapport 1, Inventaire
--> L'éditeur d'étiquettes de caractères
--> Préférences du document

Quelques-unes de ces fenêtres ne sont pas des fenêtres de composant!

La sélection

L'application et les fenêtres ont plusieurs propriétés. Particulièrement importante pour naviguer au sein d'un document est la propriété "selection".

Le contenu de la propriété "selection" d'une application correspond à l'objet sélectionné par l'utilisateur dans le document actif, lui-même placé dans une fenêtre active. Le contenu de la propriété "selection" d'une fenêtre correspond à l'objet que l'utilisateur avait sélectionné dans celle-ci, même si cette fenêtre n'est pas active (au premier plan).

```
tell application "RagTime 5"
    get selection
end tell
```

ou bien

```
tell application "RagTime 5"
    get selection of window 2
end tell
```

Exemples de réponse:

--> insertion point after character 24 of contents of text flow "Texte 1" of document id 1 of application "RagTime 5"
--> rectangle 1 of page 1 of layout "Maquette de feuillet 1" of document id 1 of application "RagTime 5"
--> cell "A1" of table "Feuille de calcul 1" of document id 1 of application "RagTime 5"

Important: la propriété "selection" est défini pour l'application et pour les fenêtres. Elle n'existe pas pour un document!

```
tell application "RagTime 5"
    tell window 1
        get selection
    end tell
end tell
```

marchera, mais:

```
tell application "RagTime 5" -- Cela ne marche pas!
    tell document 1
        get selection
    end tell
end tell
```

renverra un message d'erreur!

Notez que la valeur de la propriété "selection" est une référence complète sur des objets. Par conséquent, le code suivant a une syntaxe correcte:

```
tell application "RagTime 5"
    tell window 1
        set mySelection to selection
    end tell
    tell document 1
        -- du code ici
        delete mySelection
        -- du code ici
    end tell
end tell
```

Composants

Un document RagTime a une structure identique à celle créée par un système de gestion de fichier sur disque dur. Pour modifier un objet particulier, il est nécessaire de connaître l'adresse du composant qui le contient.

Les composants sont, par exemple:

le feuillet -- un ensemble de pages,
la maquette de feuillet,
du texte -- un composant Texte,
un tableau -- une feuille de calcul,
un composant Dessin,
une image,
un bouton.

Les feuillets sont composés de pages. Les maquettes de feuillet sont composées de pages modèles. Techniquement, ils sont un peu similaire à des composants du type Dessin.

Les scripts AppleScript sont souvent de la forme:

```
tell application "RagTime 5"
    tell document 1
        tell text flow "Texte 1"
            -- ici vient votre code
        end tell
    end tell
end tell
```

ou bien:

```
tell application "RagTime 5"
    tell document 1
        tell layout 1
            tell page 2
                -- ici vient votre code
            end tell
        end tell
    end tell
end tell
```

De même, les composants peuvent être adressés par nom ou un numéro d'index. L'index dépend de l'ordre dans lequel ils ont été physiquement stocké dans le fichier du document. Tant que votre script ne supprime pas de composants, leur numéro d'index restera le même pendant l'exécution du script.

Les composants sont créés en utilisant la commande "make". Dans la plupart des cas, la commande "make" a besoin de savoir où placer l'objet - même s'il n y a qu'une seule possibilité. Cette information est de la forme "at beginning", "at end", "at after", etc. Une autre section traite des cas où l'information sur la future position de l'objet peut être omise.

```
tell application "RagTime 5"
    tell document 1
        make new table at end
    end tell
end tell
```

Ce script créé une nouvelle feuille de calcul dans le document actif. Il n'est pas installé dans un contenant. Rien ne se passe à l'écran si vous exécutez le script sauf si l'inventaire est ouvert. La feuille de calcul devrait tout simplement apparaître dedans.

En général, les composants sont créés à la position "at end". Avec les pages, les possibilités de positionnement sont plus flexibles:

```
tell application "RagTime 5"
    tell layout 1 of document 1
        make new page at after page 1
    end tell
end tell
```

Cela marche aussi avec les positions: "at end", "at before page 1".

Les composants Dessin et les pages

Les composants Dessin et les pages sont composés d'objets de dessin. Cela peut être des rectangles, des courbes de Bézier, des textes graphiques, etc.

Les objets graphiques peuvent être référencés par le numéro d'index ou bien par le nom si il existe. Le nom peut être défini par l'utilisateur dans la palette "Coordonnées des objets". Si vous créez un objet avec un script, vous pouvez le nommer à ce moment là.

Le numéro d'index d'un objet est toujours la position relative de l'objet dans une pile d'objet. L'ordre de la pile d'objet correspond à l'ordre "visuel" des objets. l'objet d'index 1 est celui qui peut chevaucher tous les autres objets. Cet objet est au sommet de la pile, c'est à dire affiché au premier plan.

Les objets Dessin sont regroupés dans des sous-classes:

les rectangles
les lignes
les textes graphiques
les arcs
les ovales
les secteurs
les polygones
les courbes de Bézier
les multigones

Le numéro d'index d'un objet dans sa sous-classe peut être un nombre plus petit que celui si on se place dans la pile qui confond toutes les sous-classes: l'objet "polygon 1" peut être l'objet de Dessin numéro 5. Dans ce cas, cela signifie qu'il y a 4 objets appartenant à d'autres sous-classes au dessus de ce polygone.

```
tell application "RagTime 5"
    tell document 1
        delete rectangle 1 of page 1 of layout 1
    end tell
end tell
```

Ce script supprimera un rectangle de la première page du feuillet 1. Il s'agira d'un rectangle placé au dessus de tous les autres rectangles. “delete drawing object 1” supprimerait l'objet situé au sommet de la pile principale (celle qui confond toutes les sous-classes). Cela pourrait être un rectangle, un ovale ou un texte graphique.

Les objets Dessin sont créés avec la commande “make”. Il est possible de communiquer une position absolue ou une position relative:

```
tell application "RagTime 5"
    tell document 1
        make new rectangle at beginning of page 1 of layout 1
    end tell
end tell
```

Le rectangle créé sera au dessus de tous les autres objets, tout comme si l'utilisateur l'avait créé en utilisant l'outil Rectangle. Utilisez la position "at end" pour le créer derrière tous les autres objets (cela est possible seulement à partir d'un script, mais pas via l'interface utilisateur de RagTime).

Utilisez les expression "at before" ou "at after" pour le créer dans une position par rapport à un objet existant:

```
tell application "RagTime 5"
    tell page 1 of layout 1 of document 1
        make new rectangle at after drawing object 1 with data {100, 100, 200, 300}
    end tell
end tell
```

Cela créera un rectangle juste derrière l'objet qui se trouve au sommet de la pile principale. Ce script fixe les coordonnées du rectangle avec des valeurs mesurées en point (ou 1/72 pouce).

Coordonnées des objets Dessin

Les coordonnées sont mesurées en point (=1/72 pouce). L'origine du repère est le coin supérieur gauche de la page ou du composant Dessin dans lequel sera placé l'objet. Contrairement au système de coordonnées des mathématiques, le Macintosh oriente l'axe des ordonnées (y) de bas en haut.

Quand vous créez un rectangle, les coordonnées peuvent être immédiatement fournis en paramètres:

```
tell application "RagTime 5"
    tell layout 1 of document 1
        make new rectangle at beginning of page 1 with data {100, 100, 300, 200}
    end tell
end tell
```

Dans la liste, l'ordre des coordonnées est: {X gauche, Y haut, X droite, Y bas}.

Vous rencontrerez deux types de coordonnées dans le dictionnaire AppleScript: ceux du rectangle de définition (definition rect) et ceux du rectangle de guide (bound). Cela semble redondant, mais en fait non:

RagTime permet de transformer les objets: rotation, mise à l'échelle et inclinaison.

Le rectangle de définition est l'ensemble des coordonnées avant qu'une quelconque transformation ne soit appliquée. Le rectangle de guide est le plus petit rectangle qui enveloppe l'objet lorsqu'il est sélectionné.

Les propriétés "left" et "left position" se comportent un peu de la même manière. La seconde forme est la valeur de l'objet après transformation.

Faisons quelques expérimentations. Dans le script ci-dessous, les réponses d'AppleScript sont marquées par “-->”. Elles ont été copié de la fenêtre "Historique des événements" de l'éditeur de scripts d'AppleScript:

```
tell application "RagTime 5"
    make new rectangle at beginning of page 1 of layout 1 of document 1¬
        with data {200, 250, 300, 400}
    --> rectangle 1 of page 1 of layout "Feuillet 1" of document id 2
    set vertical scaling factor of rectangle 1 of page 1 of layout 1 of document 1 to 2.5
    set horizontal scaling factor of rectangle 1 of page 1 of layout 1 of document 1 to 2.5
    get definition rect of rectangle 1 of page 1 of layout 1 of document 1
    --> {200.0, 250.0, 300.0, 400.0}
    get bounds of rectangle 1 of page 1 of layout 1 of document 1
    --> {125.0, 137.5, 375.0, 512.5}
    get left of rectangle 1 of page 1 of layout 1 of document 1
    --> 200.0
    get left position of rectangle 1 of page 1 of layout 1 of document 1
    --> 125.0
end tell
```

"bounds" donne les plus grandes valeurs et "left position" correspond plus à "à la gauche" parce que la mise à l'échelle de 250% est comptée dedans.

Création de polygones et de courbes de Bézier

Dans RagTime, il est possible, via un script, de créer des formes géométriques sophistiquées d'objets en se basant sur les polygones ou les courbes de Bézier.

En général, de telles courbes sont définies par une liste de points. Vous pouvez soit créer une courbe en affectant une telle liste de points à l'une de ses propriétés, soit en modifiant la liste de points d'un objet déjà existant. Cela changera la forme et la position de la courbe.

Pour bien comprendre ce concept, il suffit de dessiner une courbe dans RagTime et de lui demander la liste de points de cette courbe.

Dans un premier exemple, j'ai un polygone constitué de trois segments, quelque part sur une page. Le polygone est sélectionné. Le script suivant:

```
tell application "RagTime 5"
    get point list of selection
end tell
```

retourne une valeur comme ci-dessous:

{{77.76, 275.76}, {231.84, 201.6}, {326.16, 308.16}, {447.12, 207.36}}

C'est une liste de listes. La liste principale est constituée d'éléments qui correspondent à des points

(sommet) du polygone (il y a donc 4 éléments). Ces éléments sont composés de deux valeurs: il s'agit de l'abscisse x et de l'ordonnée y de ces points placés sur la page.

Ce même script est exécuté alors qu'une courbe de Bézier est sélectionnée. Là aussi, la courbe est constituée, par hypothèse, de trois segments:

{{69.84, 283.68}, {133.2, 252.72}, {193.68, 185.04}, {257.76, 241.2}, {321.84, 297.36}, {319.68, 379.44}, {388.8, 339.84}, {457.92, 300.24}, {491.76, 239.76}, {491.76, 239.76}}

La structure est la même, mais il y a plus de points (10 au total). Alors que l'idée d'une liste de points pour décrire un polygone est assez simple, quelques explications sont nécessaires concernant les courbes de Bézier (cette explication n'est pas lié à RagTime, mais plutôt à la définition ce type de courbe).

En termes mathématiques, les courbes que vous dessinez avec les divers outils de ligne de Bézier dans une application sont des séquences de courbes de Bézier de base. Dans ce cadre stricte, une courbe de Bézier est définie par exactement quatre points: le point de départ, le point d'arrivée et deux points de tangente. Les deux points de tangente indiquent dans quelle direction la courbe quitte le point de départ et le point d'arrivée. De plus, ces points de tangente définissent la courbature au niveau du point de départ et du point d'arrivée.

Ces séquences de courbes de Bézier, quand vous les dessinez, sont toujours interconnectées. Le premier morceau partage le point d'arrivée avec le point de départ du morceau suivant, et ainsi de suite.

Pour la liste de points, cela signifie: vous avez besoin de quatre points pour le premier morceau mais seulement de trois points pour tous segments supplémentaires. Le nombre de points dans une séquence de courbes de Bézier est toujours 4 + n * 3 où n est un nombre entier (0 inclut).

Revenons à RagTime: le premier script dessinera un carré mais ouvert sur le côté supérieur. La longueur d'un côté est de 1 pouce (2,54 cm). Le point supérieur gauche sera placé à 1 pouce du bord supérieur gauche de la page.

```
tell application "RagTime 5"
    set myPointList to {{72, 72}, {72, 144}, {144, 144}, {144, 72}}
    tell page 1 of layout 1 of document 1
        make new polygon at beginning with properties ¬
            {point list:myPointList}
    end tell
end tell
```

Exécutez à nouveau le même script après avoir remplacé "make new polygon" par "make new Bezier curve". Le résultat sera une courbe qui ressemble à une moitié d'un ovale contenu dans le polygone ouvert précédemment créé.

Pour modifier une courbe existante utilisez tout simplement "set point list of ... to ...".

Grouper des objets

Les objets Dessin peuvent se constituer en groupe. Dans le dictionnaire AppleScript de RagTime, un groupe est un objet défini. Pour grouper des objets, vous devez d'abord créer un objet du type groupe et lui indiquer ses futurs membres:

```
tell application "RagTime 5"
    tell page 1 of layout 1 of document 1
        make new drawing group at end with data {rectangle 1, rectangle 2, rectangle 3}
    end tell
end tell
```

Ce script créera un nouvel objet du type groupe, composé des trois rectangles de la page 1. Il s'agit des rectangles situés au sommet de la pile des objets de la sous-classe des rectangles.

Pour dégrouper, il y a une commande spéciale:

```
tell application "RagTime 5"
    tell page 1 of layout 1 of document 1
        ungroup drawing group 1
    end tell
end tell
```

Les objets Dessin et les composants

Les rectangles et les autres types d'objets de dessin d'une page RagTime sont souvent utilisés pour contenir des composants: texte, feuille de calcul, image, etc. Voici quelques méthodes à suivre pour les créer:

```
tell application "RagTime 5"
    tell page 1 of layout 1 of document 1
        make new rectangle at beginning with data {200, 250, 300, 400} ¬
            with properties {contents type:table}
    end tell
end tell
```

C'est le plus court chemin pour créer un rectangle contenant une feuille de calcul (précisons que la liste de propriétés peut contenir plus de caractéristiques comme le format de filet, la couleur, etc).

L'inconvénient de cette technique est la suivante: vous ne gardez pas en mémoire une référence sur le tableau récemment créé. Voici une alternative au script précédent:

```
tell application "RagTime 5"
    tell document 1
        set newTable to (make new table at end)
        tell page 1 of layout 1
            make new rectangle at beginning with data {200, 250, 300, 400}
            set contents of rectangle 1 to newTable
        end tell
    end tell
end tell
```

Ici, la variable "newTable" contient une référence sur la feuille de calcul fraîchement créée et peut être utilisée par la suite pour l'éditer. Le contenu de la variable"newTable" sera une référence de la forme:

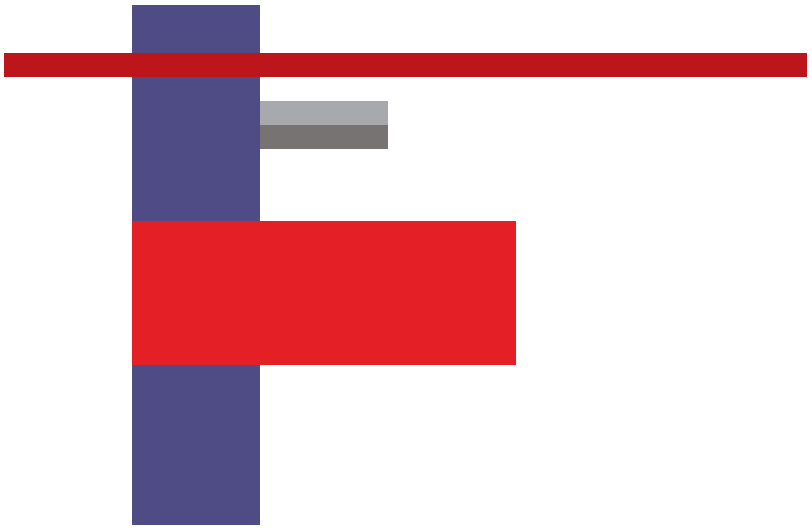
--> table "Feuille de calcul 1" of document id 1

Les tableaux

Les tableaux sont des objets avec une structure interne claire et concise. Ainsi, ils sont facilement contrôlables à partir d'un script. Ils sont composés de colonnes, de lignes, de rangées et de cellules.

Le script suivant peint une rangée de cellule dans différentes couleurs. Créez un nouveau document avec un seul tableau et exécutez le script pour observer le résultat.

```
tell application "RagTime 5"
    tell table 1 of document 1
        set color of column 2 to {cyan:80, magenta:80, yellow:20, black:5} -- bleu foncé
        set color of row 3 to {cyan:5, magenta:100, yellow:100, black:20} -- rouge foncé
        set color of cell 5 of column 3 to {cyan:0, magenta:0, yellow:0, black:40} -- gris clair
        set color of cell "C6" to {cyan:20, magenta:20, yellow:20, black:50} -- gris foncé
        set color of range "B10:D15" to {cyan:5, magenta:100, yellow:100, black:0} -- rouge
    end tell
end tell
```



Cette feuille de calcul a été créé par le précédent script

Si votre feuille de calcul a plusieurs plans, les cellules devront être adressées par la chaîne de caractère qui décrit le plan et la rangée:

...
set color of range "[3]B:[3]B" to {cyan:80, magenta:80, yellow:20, black:5}
..

Cela modifie la couleur de la colonne B du plan 3.

Les cellules: valeur, formule et format des valeurs

Les principales propriétés d'une cellule sont sa valeur et sa formule.

```
tell application "RagTime 5"
    tell column 1 of table 1 of document 1
        set value of cell 1 to "abc"
        set value of cell 2 to 250.052
    end tell
end tell
```

Cela met “abc” dans la cellule A1, et 250.052 dans la cellule A2.

Le script suivant assigne la même formule à deux cellules mais d'une manière différente:

```
tell application "RagTime 5"
    tell column 1 of table 1 of document 1
        set formula of cell 1 to "Somme(B:B)"
        set sylk formula of cell 2 to "SUM(C[1]:C[1])"
    end tell
end tell
```

Les deux cellules A1 et A2 auront par la suite la formule “Somme(B:B)”. Cela fonctionne à condition que vous utilisez une version française de RagTime. Sinon, l'instruction "set formula of cell 1 to "Somme(B:B)"" retournera une erreur. Supprimer l'instruction et essayez le reste. La mise en place de la formule au format SYLK fonctionnera dans n'importe quelle version linguistique de RagTime.

Comment retrouver la syntaxe SYLK d'une formule: pour cela, utilisez un script.

```
-- Script auxiliaire pour se familiariser avec la syntaxe SYLK
tell application "RagTime 5"
```

```
    set theSYLKFormula to (sylk formula of selection) as text
end tell
set the clipboard to theSYLKFormula
```

Sélectionnez une cellule qui a une formule et exécutez le script. La version SYLK de la formule sera affichée dans une zone de dialogue.

Saisie de valeurs et reconnaissance automatique du type de valeur

Lorsque vous saisissez une valeur dans une cellule de feuille de calcul, RagTime essaie de reconnaître le type de valeur: un nombre, une date, une durée, un texte multiligne, etc.

La chaîne de caractère "2/15/2002" sera interprétée comme une date si cela correspond aux préférences de votre système d'exploitation.

Si vous utilisez l'expression "value of" comme vu dans un script précédent, aucune reconnaissance n'aura lieu.

set value of cell "A2" to "2/15/2002"

insérera la chaîne de caractère comme texte sans provoquer la reconnaissance automatique.

set cell “A2” to “2/15/2002”

insérera une date comme si la chaîne avait été saisie manuellement: la reconnaissance automatique aura lieu.

Exemple:

```
tell application "RagTime 5"
    tell column 1 of table 1 of document 1
        set cell 1 to "15%"
        -- A1 est maintenant le nombre 0,15. Le format est mis à "Pourcentage".
        set value of cell 2 to "15%"
        -- A2 contient le texte "15%"
    end tell
end tell
```

Utiliser "value of" est très direct et plus rapide. Si vous devez entrer une longue série de valeurs (une boucle par exemple), demandez-vous si la reconnaissance automatique est nécessaire ou non.

Accéder aux textes

Un composant Texte est appelé “text flow” dans le dictionnaire AppleScript de RagTime. Il est composé de paragraphes, de mots et de caractères.

Le script suivant met la totalité d'un composant Texte dans une variable AppleScript:

```
tell application "RagTime 5"
    tell document 1
        set myText to text flow 1 as text
    end tell
end tell
```

Notez que l'expression "as text" change le type de l'objet (coercition). Il est nécessaire ici d'obtenir des informations venant de RagTime dans un type correct.

Le résultat est un texte en style normal. Utilisez "as styled text" pour placer le texte stylé dans la variable tout en préservant les différents styles de caractère qu'il contient.

Les autres manières typiques pour accéder à du texte sont:

```
tell application "RagTime 5"
    tell text flow 1 of document 1
        set myText to paragraphs 2 thru 6 as text
    end tell
end tell
```

Observez le script suivant:

```
tell application "RagTime 5"
    tell text flow 1 of document 1
        set myText to (words 2 thru 6) as text
    end tell
end tell
```

Probablement, le résultat retourné vous surprendra: tous les mots seront renvoyés par RagTime sous la forme d'une liste. AppleScript autorise la transposition de type (coercition) “as text” sur une liste, il en résulte une chaîne de caractère où tous les mots sont concaténés (cela dépend du caractère délimiteur courant d'AppleScript). Pour obtenir le vrai texte, procédez comme suit:

```
tell application "RagTime 5"
    tell text flow 1 of document 1
        set myText to text from word 2 to word 6 as text
    end tell
end tell
```

"text" est une propriété spéciale des textes pour éviter ce problème.

Écrire dans les composants Texte

Une manière simple d'ajouter du texte à un composant texte est d'utiliser l'expression "make new text":

```
tell application "RagTime 5"
    tell text flow 1 of document 1
        make new text at end with data " le mot de la fin"
    end tell
end tell
```

La chaîne de caractère " le mot de la fin" sera ajoutée au composant existant "text flow 1". D'autres positions peuvent être spécifiées: "at beginning" (placer au début), "at after" (à la suite de), et "at before" (placer avant).

Exemples:

```
tell application "RagTime 5"
    tell text flow 1 of document 1
        make new text at after paragraph 2 with data ¬
            "Ceci est un paragraphe supplémentaire après le paragraphe 2\r"
    end tell
end tell
```

Remarquez que l'expression “\r”, pour retour chariot, est seulement utilisée à la fin. L'expression “after paragraph” placera le texte au début du paragraphe suivant, ce dernier étant décalé d'un paragraphe.

En travaillant avec du texte, assez souvent, vous souhaiterez insérer du texte au lieu de remplacer. En d'autres termes, parfois, vous voudrez insérer "xyz" entre le caractère 2 et le caractère 3. Cela ne peut pas être fait en remplaçant un caractère particulier par autre chose. L'outil pour accomplir cela est l'expression "insertion point", commande standard d'AppleScript pour la manipulation de textes. L'exemple suivant insère un peu de texte au début du paragraphe 2:

```
tell application "RagTime 5"
    tell text flow 1 of document 1
        set insertion point before paragraph 2 to ¬
            "Ceci est la nouvelle et première phare du paragraphe 2. "
    end tell
```

```
end tell
```

L'effet est le même si vous remplacez "before paragraph 2" par "after paragraph 1". Les deux formes insèrent au début d'un paragraphe mais ne créées pas un nouveau paragraphe. Pour ajouter à la fin d'un paragraphe, utilisez l'instruction:

...
set insertion point after character -2 of paragraph 2 to...
...

Notez que l'expression "insertion point" est utilisée avec "after" (après) et "before" (avant) mais pas avec "at" (à): le point d'insertion n'est pas un objet défini.

Formules dans le texte

Une formule est considérée par RagTime comme une propriété. Cela paraît logique s'il s'agit de cellules d'une feuille de calcul mais rend parfois perplexe les gens lorsqu'il s'agit d'appliquer cette idée à l'objet texte. Pour avoir un texte calculé dans RagTime, vous n'insérez pas une formule dans le texte mais plutôt vous activez la propriété formule à un endroit du texte.

Habituellement, la propriété formule sera appliquée au point d'insertion. Si vous le désirez, un format de valeurs peut être également spécifié, par exemple en l'appliquant au premier caractère de l'expression mathématique récemment insérée.

```
tell application "RagTime 5"
    tell text flow 1 of document 1
        set insertion point before paragraph 2 to " "
        set formula of insertion point before paragraph 2 to "AujourdHui()"
        set value format of first character of paragraph 2 to "Date longue"
    end tell
end tell
```

Cela insère une date automatique au début du paragraphe 2, suivit d'un espace. La numérotation de paragraphes dans le texte se met en place de la même façon:

```
tell application "RagTime 5"
    tell text flow 1 of document 1
        set insertion point before paragraph 2 to "\t"
        set paragraph numbers of insertion point before ¬
            paragraph 2 to "<R+>"
    end tell
end tell
```

Ce code insère un numéro de paragraphe (en chiffre romain) au début du paragraphe 2. Il est suivi d'une tabulation (la syntaxe pour les formules de numérotation de paragraphe est décrite dans le manuel de référence de RagTime, dans le chapitre "Texte", paragraphe "Numérotation des paragraphes").

Travailler avec des images

Importation d'images

Le code qui suit importe dans un document RagTime une image contenue dans un fichier:

```
set myPicture to choose file with prompt "Choisissez une image"
tell application "RagTime 5"
    tell document 1
        make new picture at end with data myPicture
    end tell
```

```
end tell
```

Le composant Image apparaîtra dans l'inventaire mais il n'est pas placé quelque part sur une page ou dans un objet de Dessin. Pour remplacer le contenu d'un composant image existant par une autre image provenant d'un fichier, utilisez le script suivant:

```
set myPicture to choose file with prompt "Choisissez une image"
tell application "RagTime 5"
    tell document 1
        change picture 1 to myPicture
    end tell
end tell
```

Lorsque vous utilisez la commande "change", les propriétés comme "linked to file" (lié au fichier) peuvent être spécifiées:

```
set imageFile to choose file with prompt "Choisissez une image"
tell application "RagTime 5"
    tell document 1
        set myPicture to make new picture at end
        change myPicture to imageFile with link to file
    end tell
end tell
```

Utilisez l'instruction "...without link to file" (ne pas lier au fichier) pour importer explicitement la totalité du fichier. Je reviendrai sur l'utilisation plus générale de l'instruction "change" dans une autre section.

Mettre à l'échelle des images

Les images possèdent des propriétés de mise à l'échelle, échelle horizontale et échelle verticale. Cela peut paraître naturel. Si vous vous êtes bien familiarisés avec les concepts de RagTime, c'est plutôt surprenant:

Une image peut être placée plusieurs fois dans un document. Dans RagTime, cela ne signifie pas nécessairement que l'image est dupliquée autant de fois. C'est à dire, l'image n'apparaît qu'une seule fois dans l'inventaire, mais apparaît dans plusieurs contenants. L'échelle peut être différente d'un contenant à l'autre. En d'autre terme, l'échelle s'applique à une installation spécifique de l'image dans un contenant.

Cette approche n'est peut-être pas celle d'un débutant en train de travailler avec RagTime. Mais pour le pilotage, c'est une intéressante fonctionnalité. Cette section souligne la façon de mener à bien des installations multiples d'une même image.

Faisons quelques expérimentations. Deux images ont été importé dans un document. La première est placée dans un contenant et mise à une échelle de 50%. La deuxième image est placée dans deux contenants, l'un avec une échelle de 25% et l'autre avec une échelle de 75%. Voici les réponses AppleScript de RagTime que la fenêtre d'historique des événements rapporte lorsqu'il est demandé l'échelle de ces images:

```
tell application "RagTime 5"
    tell document 1
        get vertical scaling factor of picture 1
        --> 0.5
        get vertical scaling factor of picture 2
        --> {0.25, 0.75}
    end tell
end tell
```

("vertical": il doit y avoir par conséquent une échelle horizontale).
Si le résultat ne renvoie qu'un seul nombre, alors cela signifie que l'image est placée qu'une seule fois. Dans le cas de la seconde image, le résultat est une liste contenant toutes les échelles utilisées. (Un nombre seulement est retourné lorsque l'image est placée dans différents contenants montrant l'image à

la même échelle!)

Seul des nombres sont autorisés pour modifier une échelle:

```
tell application "RagTime 5"
    tell document 1
        set scaling factor of picture 1 to 0.5
    end tell
end tell
```

Cela met l'échelle de l'image 1 à 50% pour tous les contenants où l'image est installée. (En utilisant l'expression "scaling factor" au lieu de "vertical scaling factor", l'échelle est modifiée dans les deux directions).

Obtenir ou changer l'échelle dans un contenant bien particulier peut être possible grâce à la propriété "contents" de ce contenant:

Pour avoir l'information recherchée:

```
tell application "RagTime 5"
    tell page 1 of layout 1 of document 1
        vertical scaling factor of contents of rectangle 2
    end tell
end tell
```

Pour modifier:

```
tell application "RagTime 5"
    tell page 1 of layout 1 of document 1
        set scaling factor of contents of rectangle 2 to 0.3
    end tell
end tell
```

(Évidemment, ce script fonctionnera correctement si le rectangle 2 contient vraiment une image).

Alignement d'images

Aligner une image via un script est assez proche de la façon de faire via l'interface utilisateur:

```
tell application "RagTime 5"
    tell document 1
        align picture picture 1 relative size fitting picture to frame
    end tell
end tell
```

(Note: la commande est “align picture”, ce qui explique pourquoi deux mots identiques se suivent).

Ce script ajuste l'image 1 au contenant et ceci dans tous les contenants qui affichent cette image. Dans tous les cas , un message sera affiché demandant si les proportions originales doivent être conservées ? Ce qui suit est une légère modification du script:

...
align picture picture 1 relative size fitting picture to frame keeping proportions
...

afin qu'il respecte les proportions.

La commande "align picture" ne peut pas s'appliquer à la propriété "contents" du contenant comme nous l'avons vu dans la section concernant la mise à l'échelle. Le script qui suit ne marchera pas comme prévu:

```
tell application "RagTime 5"
    tell page 1 of layout 1 of document 1
        align picture (contents of rectangle 1) horizontally at left side
    end tell
end tell
```

Il alignera l'image dans tous les contenants puisque l'instruction "contents of rectangle 1" est résolu en premier !

C'est l'un des rares cas où vous devez faire une sélection lorsque vous pilotez RagTime:

```
tell application "RagTime 5"
    tell page 1 of layout 1 of document 1
        select contents of rectangle 1
    end tell
    align picture selection horizontally at left side
end tell
```

(Avertissement: les documents ne savent pas ce qu'est une sélection, l'application et les fenêtres le savent. C'est la raison pour laquelle la commande "align picture" doit être en dehors du bloque "tell document...end tell". Voir le début de ce document.).

Créer des graphes

Les graphes ont beaucoup de propriétés. Heureusement, la plupart d'entre elles sont facile à utiliser. Pour éviter toute ambiguïté dans cet exemple, le script créé la totalité du document. Premièrement, une feuille de calcul avec quelques valeurs dedans est créée. Les valeurs sont simplement insérées comme texte. Cela revient à les saisir manuellement avec la reconnaissance automatique du format de valeurs en action. Deuxièmement, un graphe est créé et lié à la feuille de calcul.

```
----------
set values1 to "4
6
3
8"
set values2 to "2
3
9
3"
tell application "RagTime 5"
    set myDocument to (make new document with properties {component types:layout} at end)
    tell myDocument
        delete every drawing object of page 1 of layout 1
        set myTable to (make new table at end)
        tell myTable
            set range "A2:A6" to values1
            set range "B2:B6" to values2
        end tell
        make new rectangle at beginning of page 1 of layout 1 with data {100, 50, 400, 250}
        set contents of rectangle 1 of page 1 of layout 1 to myTable
        make new rectangle at beginning of page 1 of layout 1 with data {100, 300, 400, 800}
        -- Ici, la page est prête et la création du graphe commence
        set myGraph to (make new graph at end)
        set contents of rectangle 1 of page 1 of layout 1 to myGraph
        set chart type of myGraph to xy graph
        set newSeries to (make new series at end of myGraph)
        set formula of x value list of newSeries to name of myTable & "!A2:A7"
        set formula of y value list of newSeries to name of myTable & "!B2:B7"
    end tell
end tell
----------
```

Le point le plus intéressant ici est de voir comment les formules sont créées à partir d'une série de données. La formule est d'abord créée comme texte. Ensuite, ce texte devient une propriété "formula" de la série.

Chaînage de feuillet, chaînage de maquette de feuillet

Les chaînages de feuillet sont des éléments d'un document placés au sommet de la hiérarchie des objets. Il n'appartiennent à aucun type de composant. Les chaînages de maquette sont d'un type légèrement différent. Eux aussi ne font pas partie d'une famille de composant, mais sont des éléments placés au sommet de la hiérarchie.

Les chaînages sont créés à partir d'une liste de références sur des contenants. Voici un exemple de script qui illustre bien ce concept de chaînage. Nous considérons un document avec un seul feuillet de plusieurs pages. Sur chaque page, il y a un rectangle vide, sans chaînage entre eux. Les premiers rectangles de ces pages doivent être chaînés.

```
set rectList to {}
tell application "RagTime 5"
    tell document 1
        tell layout 1
            repeat with n from 1 to count pages
                copy (a reference to rectangle 1 of page n) to end ¬
                    of rectList
            end repeat
        end tell
        set myPipeline to make new pipeline with data rectList at end
    end tell
end tell
```

Lorsque la boucle (loop) se termine, la variable "rectList" contient toutes les références sur les rectangles dans le bon ordre. Le chaînage est généré à partir de cette liste en utilisant la commande "make". La variable "myPipeline" reçoit une valeur du style:

pipeline "Chaînage 1" of document id 1 of application "RagTime 5"

Si vous envisagez de modifier ce chaînage un peu plus tard dans le script, il est bon de garder en mémoire (grâce à une variable) une référence sur ce chaînage.

Modifier les chaînages

Le script suivant créé un chaînage qui connecte des rectangles placés sur des pages paires grâce à un chaînage vertical. Chacun de ces contenants est relié par un chaînage horizontal à la page impaire qui suit. Cela n'a de sens que pour des tableaux d'une brochure.

Élaborer un tel chaînage pour des feuilles de calcul signifie que les premières colonnes du tableau apparaîtront sur les pages 2, 4, 6... Les colonnes supplémentaires apparaîtront sur les pages 3, 5, 7... (Si vous ouvrez un livre, la page de gauche a un numéro de page paire, et la page de droite a un numéro de page impaire).

```
set rectList to {}
tell application "RagTime 5"
    tell document 1
        tell layout 1
            set pageCount to count pages
            repeat with n from 2 to pageCount by 2
                copy (a reference to rectangle 1 of page n) to end ¬
                    of rectList
            end repeat
        end tell
        set myPipeline to make new pipeline with data rectList at end

        repeat with n from 1 to round ((pageCount - 1) / 2) rounding down
            set horizontal successor of pipeline element n of myPipeline ¬
                to rectangle 1 of page (n * 2 + 1) of layout 1
```

```
        end repeat
    end tell
end tell
```

La première boucle créée le chaînage vertical comme dans le précédent script. La deuxième boucle édite ce chaînage en y ajoutant de nouveaux contenants mais avec une liaison horizontale.

Les propriétés "horizontal successor", "vertical successor", "horizontal predecessor" et "vertical predecessor" appartiennent à l'objet "pipeline element". Elles peuvent être utilisées pour étendre un chaînage à n'importe quel endroit ou bien pour le modifier.

Les étiquettes de style

D'une manière générale, une étiquette est une collection nommée d'attributs. Cette collection peut être appliquée à des objets en un seul geste. Toute modification d'une étiquette se répercutera sur les objets du document où l'étiquette a été appliqué. Une étiquette de caractère possède des propriétés comme la police, le corps, la couleur, etc. Une étiquette de paragraphe contient des informations sur les marges, les distances entre les lignes et entre les paragraphes, etc. Une étiquette de filet regroupe des propriétés comme l'épaisseur, le type de pointillé, la forme des extrémités, etc. Techniquement, les règles et les unités se comportent aussi de la même manière.

Les étiquettes se trouvent au sommet de la hiérarchie des objets du document. En travaillant dans RagTime, un utilisateur a accès à deux groupes d'étiquettes: les étiquettes qui sont dans le document et les étiquettes stockées dans le fichier des accessoires de RagTime.

Chaque nouvel objet est créé en s'appuyant sur certaines étiquettes standards. Leurs propriétés s'inspirent de celles des accessoires de RagTime. C'est pour cette raison que l'environnement de travail de RagTime est en partie définit par les étiquettes des accessoires de RagTime. Pour citer l'exemple le plus connu, l'étiquette de caractère "Caractère standard" définit la police et le corps par défaut de tout nouveau document.

Les étiquettes de caractère peuvent être appliquées sur des objets via un script simplement en modifiant la propriété "style sheet" d'un objet. Normalement, vous ferez cela en utilisant le nom de l'étiquette.

Quelques précautions sont à prendre: utiliser un nom d'étiquette qui n'existe pas fera tout simplement planter RagTime! C'est un bug reconnu, mais pour l'instant vous devez vivre avec. Par conséquent, vérifier d'abord son existence!

Le script suivant montre comment solliciter une étiquette de caractère avec précaution, qu'elle soit dans les accessoires de RagTime ou dans le document courant. Si une étiquette de caractère appelée "En-tête" n'existe pas, rien ne se passe:

```
tell application "RagTime 5"
    set headerStyle to "En-tête"
    tell auxiliaries document
        -- ce qui suit vérifie l'existence de l'étiquette de caractère dans les accessoires de RagTime,
        -- où résident toutes les étiquettes prédéfinies.
        set headerStyleExists to (exists character style sheet "En-tête")
    end tell
    tell document 1
        set headerStyleExists to ((exists character style sheet headerStyle) or headerStyleExists)
        -- Si l'étiquette existe seulement dans le document courant, c'est tout aussi bien
        tell text flow 1
            if headerStyleExists then
                set character style sheet of paragraph 2 to headerStyle
            end if
        end tell
    end tell
end tell
```

Alors qu'il existe différent type d'étiquettes, la façon de les manipuler dans un script est assez classique.

```
tell application "RagTime 5"
    tell document 1
        make new character style sheet at end
    end tell
end tell
```

créé une nouvelle étiquette de caractère. Elle portera un nom standard (dans la version française de Ragtime, le nom sera "Sans titre", alors que dans la version anglaise, le nom sera "Untitled"). Ce script provoquera l'ouverture de l'éditeur d'étiquettes de caractère, prêt à paramétrer l'étiquette.

L'éditeur d'étiquettes ne sera pas ouvert si l'étiquette est créée à partir de ses propriétés:

```
tell application "RagTime 5"
    tell document 1
        make new character style sheet with properties ¬
            {name:"Ètiquette de caractère  1", font:"Arial", size:10} at end
    end tell
end tell
```

Sans changement visuel dans les fenêtres de RagTime (sauf dans l'inventaire si la liste des étiquettes de caractère est visible), le document s'enrichira d'une nouvelle étiquette de caractère nommée "Étiquette de caractère 1" laquelle a pour corps 10 points et pour police Arial.

Hiérarchies des étiquettes de caractère

La plupart des étiquettes peuvent être organisées sous la forme de hiérarchie. Chaques étiquettes d'un niveau inférieur héritent des propriétés de son parent. Une étiquette placée sous celle nommée "Caractère standard" peut hériter de toutes les propriétés de son parent, tout en ayant, par exemple, l'attribut taille différent. Cela signifie que si quelqu'un change la police de l'étiquette de caractère intitulée "Caractère standard", tout le texte qui utilise notre nouvelle étiquette de caractère verra changer sa police. Seul l'attribut taille n'hérite pas.

Pour effectuer cela dans un script, il est nécessaire de placer une étiquette à un certain niveau dans la hiérarchie.

```
tell application "RagTime 5"
    tell document 1
        set style_1 to (make new character style sheet with properties ¬
            {name:"Style racine"} at before character style sheet 1)
        set style_2 to (make new character style sheet with properties ¬
            {name:"Enfant 1"} at end of style_1)
        set style_3 to (make new character style sheet with properties ¬
            {name:"Enfant 2"} at end of style_1)
        set style_4 to (make new character style sheet with properties ¬
            {name:"Petit-fils"} at end of style_2)
    end tell
end tell
```

"Style racine" sera créé avant la première étiquette de caractère de la série. Ce sera le sommet, aucune propriété n'est pour l'instant léguée.

Sous "Style racine", ses deux enfants "Enfant 1" et "Enfant 2" sont créés "Petit-fils" sera une progéniture de "Enfant 1".

Étiquettes de couleur

RagTime 5.5 fut la première version à intégrer la gestion et la séparation des couleurs, pour tous les objets. Depuis cette version, les étiquettes de couleur peuvent se baser sur les modèles de couleur sui-

vant: RGB, CMJN (CMYK en anglais) et CIE Lab (D50). Les étiquettes de couleur ("color" est un objet défini dans le dictionnaire AppleScript de RagTime) sont comparables aux couleurs globales d'Adobe Illustrator.

Les objets du type "color" sont créés à partir d'une structure du type enregistrement qui spécifie à la fois le modèle de couleur et la valeur des coordonnées chromatiques:

```
tell application "RagTime 5"
    tell document 1
        make new named color with properties ¬
            {name:"Rouge Lab", kind:process, color:{LabL:50, Laba:80, Labb:80}} at end
    end tell
end tell
```

(Il n'y a aucun intérêt à mélanger les modèles de couleur dans la structure "color".)

Pour le modèle Lab, les valeurs de luminosité vont de 0 à 100. Les valeurs de a et b sont compris entre -127 et +128.

Pour les modèles RGB et CMJN, les valeurs s'expriment en pour-cent, et sont compris entre 0 et 100.

Importation de fichiers

Importation avec les commandes "make" et "set"

Il est possible d'importer des données dans les documents RagTime. Le fichier sera représenté par un objet ayant pour valeur une référence d'alias ou une référence de chemin d'accès. Inversement, un objet existant peut être modifié en référence d'alias ou en référence de chemin d'accès.

```
set textFile to choose file with prompt "Choisissez un fichier texte" of type list {"TEXT"}
tell application "RagTime 5"
    activate
    tell document 1
        make new text flow with data textFile at end
        -- Cela importe le fichier texte dans un nouveau composant Texte.
        -- Ce composant n'est pas placé dans un contenant.
        set myRect to make new rectangle with data {100, 50, 300, 500} ¬
            at beginning of page 1 of layout 1
        set contents of myRect to textFile
        -- Importe le fichier et l'installe directement dans "myRect"
        set text flow 1 to textFile
        -- Importe le fichier pour remplacer le contenu courant du
        -- composant Texte référencé "text flow 1"
        set insertion point after last character of text flow 1 to textFile
        -- Importe le fichier et l'ajoute au texte "text flow 1"
        set myRect to make new rectangle with data {350, 50, 550, 500} ¬
            with properties {contents type:table} ¬
            at beginning of page 1 of layout 1
        set cell "A1" of table 1 to textFile
        -- Importe le fichier "textFile" dans une feuille de calcul en commençant par
        -- remplir la cellule A1
    end tell
end tell
```

En utilisant cette technique, tout se comporte comme une importation faite manuellement. Eventuellement, des zones de dialogues s'afficheront pour spécifier des options (encodage texte, détection automatique du format des valeurs, etc).

Importation en utilisant la commande “change”

Les options d'importation de RagTime sont sous le contrôle du script si la commande "change" est utilisée. Si du texte est importé, alors c'est une option d'encodage qui sera spécifiée:

```
set textFile to choose file with prompt "Choisissez un fichier texte" of type list {"TEXT"}
tell application "RagTime 5"
    tell document 1
        set myTextComponent to (make new text flow at end)
        change myTextComponent to textFile using encoding 256 without link to file
        set myTableComponent to (make new table at end)
        change cell "A1" of myTableComponent to textFile ¬
            using encoding 256 without link to file, value format detection and import formats
    end tell
end tell
```

Premièrement, le script créé un nouveau composant Texte et change son contenu pour celui du fichier texte. L'encodage est donné par le script.

256 signifie: Mac OS pour langues de l’Europe de l'ouest.

512 signifie: Windows pour langues de l’Europe de l'ouest.

Pour obtenir la valeur d'un encodage texte particulier, faites une importation manuelle alors qu'AppleScript est en mode enregistrement (bouton "Mémoriser")!

Ensuite, le script créé une feuille de calcul et la remplit avec le contenu du fichier texte. L'option d'encodage texte est précisée dans le script, de plus aucune détection automatique du format des valeurs n'aura lieu. Au final, la zone de dialogue des options n'apparaîtra pas durant l'importation.

Notez que ni le composant Texte, ni la feuille de calcul sont installés dans un contenant. Utilisez la commande "set" ("set contents of rectangle 1 of page 1 to...") pour le faire.

Une importation via la commande "change" permet aussi de spécifier des filtres d'importation. Cela est utile lorsqu'il s'agit d'importer des images.

...convert with "JPEG" ...

(À cause de l'absence d'une entrée dans le dictionnaire AppleEvent des versions plus anciennes n'acceptaient pas cette syntaxe mais la classe doit être déclarée entre des crochets. Si vous utilisez RagTime 5.5, faites un enregistrement pour le retrouver.)

Voici un script complet sur l'importation d'image:

```
set pictureFile to choose file with prompt "Sélectionnez une image JPEG" of type list {"JPEG"}
tell application "RagTime 5"
    tell document 1
        set pictureComponent to (make new picture at end)
        change pictureComponent to pictureFile ¬
            convert with "JPEG" without link to file
    end tell
end tell
```

Exportation

Les composants peuvent être exportés via la commande "save". Tout se passe comme s'il s'agissait de sauvegarder un document dans le propre format de RagTime.

La commande "save" fait partie de la suite standard du dictionnaire AppleScript de RagTime. Cette commande a beaucoup d'options qui permettent de spécifier les filtres d'exportations. Le script suivant exporte un composant Texte appelé "Texte 1" au format de fichier de Microsoft Word:

```
set targetFileName to (choose file name with prompt ¬
    "Nom du fichier:") as text
tell application "RagTime 5"
    save (text flow "Texte 1" of document 1) in file ¬
        targetFileName converting to "Microsoft Word 6/95"
end tell
```

Encore une fois, la meilleur façon de connaître les formats d'exportation disponible avec AppleScript est d'activer l'enregistreur AppleScript et de faire une exportation manuellement. Cela est particulièrement intéressant pour connaître la longue liste des options pour l'exportation au format PDF via Acrobat Distiller.

Scripts interactifs: opérations sur les sélections

Vous êtes dans une situation légèrement différente lorsqu'il s'agit de créer un script qui génère des documents à partir de zéro, ou lorsqu'il s'agit d'opérer sur des objets sélectionnés par l'utilisateur.

La création automatique de documents à partir d'une base de données appartient au premier cas: les données sont récupérées à partir d'une base de données (exemple: FileMaker Pro), et les pages sont créées conformément à la nature des données. Dans ce cas, votre script connaît, dès le début, la structure de votre document. Mais imaginez que vous voulez créer une ombre derrière un rectangle qui a été sélectionné. Il est nécessaire de savoir quel rectangle est sélectionné (ou: est-ce que la sélection courante est un rectangle?). Deux voies sont possibles et dépendent du cas de figure à traiter:

a) Il est possible d'imaginer un script qui ne précise pas l'objet qui sera soumis aux effets de ses opérations. La cible est inconnue. Dans un tel cas, un objet sera pris comme cible par défaut. La nature de cette cible dépendra de la dernière sélection effectuée par l'utilisateur.

b) Le script précise clairement l'objet cible.

Utilisation des "cibles inconnues"

Laisser RagTime déterminer un objet par défaut fonctionne assez bien lorsqu'il s'agit de travailler avec des objets de Dessin ou des pages. Observez le script suivant:

```
tell application "RagTime 5"
    make new rectangle with data {100, 100, 200, 300}
end tell
```

Le script ne spécifie pas l'objet (composant Dessin, feuillet, ...) dans lequel RagTime doit créer un rectangle. Le script fonctionnera correctement si l'utilisateur était en train de travailler dans un composant qui accepte de dessiner un rectangle. Dans un feuillet, si le dernier click de souris a eu lieu sur un objet de dessin contenu dans un composant Dessin alors le rectangle sera dessiné dans ce composant, ou bien si la click a eu lieu dans une zone vide d'une page, c'est elle qui recevra le rectangle. Mais le script échouera si l'utilisateur était en train de taper du texte sur cette page: dans ce cas précis, la cible est un texte or il est impossible de dessiner un rectangle dans ce type de composant.

Pour mieux comprendre ce concept de cible par défaut, le script suivant créé une ligne de guide qui passe par le milieu de l'objet courament sélectionné.

```
tell application "RagTime 5"
    set {X, Y} to position of selection
    make new vertical guide with data X
    make new horizontal guide with data Y
end tell
```

La première instruction se renseigne sur la position de l'objet sélectionné. Ensuite, les lignes de guide sont créées à partir de ces informations. Notez que pour dessiner les lignes de guide, aucun objet (une page ou composant Dessin, par exemple) n'est clairement mentionné. Aucune clause du type "at" n'est

utilisée. RagTime créera les lignes de guide dans le plan de dessin auquel appartient l'objet sélectionné. Le script suivant ne fonctionnera pas:

```
tell application "RagTime 5" -- Ce script ne fonctionne pas
    make new rectangle with data {100, 100, 200, 300} at beginning
end tell
```

Prenons un cas précis pour éclaircir cela. Considérons un feuillet de trois pages. Un rectangle de la page trois est sélectionné.

```
tell application "RagTime 5"
    tell page 1 of layout 1 of document 1
        make new rectangle with data {100, 100, 200, 200}
        make new rectangle with data {100, 100, 350, 350} at beginning
    end tell
end tell
```

Le rectangle le plus petit apparaîtra sur la troisième page, le plus large sera sur la première page. La seule différence ici, est que le rectangle le plus petit est créé avec la mention du style "at beginning" (c'est à dire "au début"). Puisque l'information sur la cible est inconnue, RagTime utilise la zone de dessin actif comme cible: il s'agit donc de la page 3 où quelque chose est déjà sélectionnée. Le petit rectangle est créé à un endroit bien précis. Il s'agit de la page 1, et c'est là qu'il apparaîtra. Notez que le bloque "tell" n'est pas suffisant pour spécifier une cible. Même si un "tell" s'adressait à "document 2", le premier rectangle aurait apparu sur la page sélectionnée de "document 1". Jetez un coup d'œil dans l'historique des événements pour le script suivant:

```
tell application "RagTime 5" -- Extrait de l'historique des événements d'AppleScript
    make new rectangle with data {100, 100, 200, 200}
    --> rectangle 1 of page 3 of layout "Feuillet 1" of document id 1
    make new rectangle with data {100, 100, 350, 350} ¬
        at beginning of page 1 of layout 1 of document 1
    --> rectangle 1 of page 1 of layout "Feuillet 1" of document id 1
end tell
```

Notez que le "ciblage" par défaut ne marche seulement que dans certaines circonstances. Le dessin est l'exemple le plus évident.

Objet sélectionné

Comme mentionné dans l'introduction, la sélection est une propriété de l'objet application et des fenêtres. Ce n'est pas une propriété de document!

Soit deux documents ouverts. Dans le document qui se trouve au premier plan, un rectangle placé sur une page est sélectionné. Dans le document en arrière plan, un peu de texte est sélectionné.

```
tell application "RagTime 5"
    set A to selection
    set B to selection of window 1
    set C to selection of window 2
    {A, B, C}
end tell
```

Compte tenu de la situation, le résultat sera plus ou moins ainsi:

{
rectangle 1 of page 1 of layout "Feuillet 1" of document id 2 of application "RagTime 5",
rectangle 1 of page 1 of layout "Feuillet 1" of document id 2 of application "RagTime 5",
text from character 1 to character 10 of contents of text flow "Texte 1" of document id 1 of application "RagTime 5"
}

Demander une référence sur la sélection à RagTime ou à la fenêtre 1 revient au même. Demander à la fenêtre 2 renvoie la sélection du deuxième document.

Maintenant, ouvrez l'éditeur d'étiquettes de caractère du premier document, sélectionnez une étiquette dans la liste et exécutez à nouveau le script:

{
character style sheet "Caractère standard" of document id 2 of application "RagTime 5",
character style sheet "Caractère standard" of document id 2 of application "RagTime 5",
rectangle 1 of page 1 of layout "Feuillet 1" of document id 2 of application "RagTime 5"
}

Ici, les deux fenêtres appartiennent au premier document. A et B sont toujours identiques, C correspond à la seconde fenêtre de ce document 1.

Sélections directes et indirectes

Cette section est principalement consacrée à un piège dans lequel vous risquez de tomber. Avec beaucoup de programmes, vous pouvez utiliser l'objet “selection” dans des commandes de script. Souvent, il est judicieux de stocker la référence sur la sélection dans une variable pour une utilisation ultérieure dans le script.

Dans la majorité des programmes, RagTime inclus, il y a une subtile différence d'ordre technique entre la sélection directe et celle stockée comme référence. Je vais essayer de vous faire comprendre cette différence grâce à des exemples. Mais le message à retenir est le suivant: si une méthode ne marche pas, essayez l'autre.

Par exemple, créez un document avec une seule page et une image. L'image est installée dans deux contenant placés sur cette page. L'image est sélectionnée dans l'un des deux contenants.

Script version 1:

```
tell application "RagTime 5"
    align picture selection horizontally at left side
end tell
```

Script version 2:

```
tell application "RagTime 5"
    set mySelection to selection
    align picture mySelection horizontally at left side
end tell
```

Le premier script modifiera l'alignement de l'image dans le contenant où elle est sélectionnée. Le second script alignera l'image dans les deux contenants.

Dans l'instruction "set mySelection to selection", la variable "mySelection" a pour valeur une référence du style "picture "Image 1" of document id 1". La commande "align" sera envoyée à ce composant. Aligner le composant prendra effet dans tous les contenants où il est installé. Le premier script travaille directement sur la sélection courante.

D'une façon générale, appliquer une commande directement sur la sélection tend à ressembler à l'application d'une commande via l'interface graphique. Mais si le doute subsiste, faites tout simplement des essais.

Grouper des objets sélectionnes

Pour créer un objet du type groupe, vous devez spécifier la position où il doit être créé. Envoyer à RagTime une commande pour créer un groupe sans rien préciser ne marche pas.

La meilleur solution est de créer un objet groupe avec la mention “at after” (à la suite de) par rapport aux objets sélectionnés. Dans un premier temps, stockez la sélection dans une variable pour avoir une référence. Cette référence vous servira pour créer le groupe.

```
tell application "RagTime 5"
    set mySelection to selection
    make new drawing group at after selection with data mySelection
end tell
```

Ce script groupe les objets sélectionnés. Cela ne veut pas dire pour autant que le groupe sera le nouvel élément sélectionné. Si vous préférez reproduire le même comportement que celui de l'interface utilisateur, où le groupe est sélectionné par la suite, utilisez le script suivant:

```
tell application "RagTime 5"
    set mySelection to selection
    set myGroup to (make new drawing group at after selection with data mySelection)
    select myGroup
end tell
```

Pour dégrouper, il y a une commande spéciale:

```
tell application "RagTime 5"
    ungroup selection
end tell
```

Ajouter ou supprimer des objets et références basées sur l'index

Dans la plupart des cas , lorsque RagTime vous renvoie une référence sur un objet, elle se basera sur un numéro d'index. Cela peut mener à des résultats inattendus. Observez les deux scripts suivant (nous considérons qu'un rectangle est sélectionné quelque part):

Version 1

```
tell application "RagTime 5"
    set mySelection to selection
    make new rectangle at before mySelection with data {50, 100, 150, 300}
    set color of mySelection to {cyan:10, magenta:50, yellow:80, black:5}
end tell
```

Version 2

```
tell application "RagTime 5"
    set mySelection to selection
    make new rectangle at before mySelection with data {50, 100, 150, 300}
    set color of selection to {cyan:10, magenta:50, yellow:80, black:5}
end tell
```

Étonnamment, le résultat de ces scripts est différent! Le premier script appliquera la couleur au nouveau rectangle récemment créé. Dans le second script, la couleur s'appliquera au rectangle sélectionné. La référence stockée dans la variable "mySelection" dans le premier script est du type "rectangle 1 of page 1 of...". C'est un index qui dépend de l'ordre d'empilement (derrière/devant) des rectangles. Si l'objet sélectionné avant le démarrage du script était "rectangle 1", il deviendra "rectangle 2" après la création avec la mention "at before" (devant) du nouveau rectangle. Puisque "mySelection" continue à utiliser le même index, il pointe sur le nouveau rectangle.

Le même phénomène se reproduit quand du texte est modifié. Concevons un petit script qui place un trait d'union avant et après le texte sélectionné.

```
tell application "RagTime 5" -- Ne marche pas comme espéré
    set mySelection to selection
    set insertion point before mySelection to "-"
    set insertion point after mySelection to "-"
end tell
```

L'insertion du premier trait d'union change le nombre de caractères. Le second trait d'union sera donc inséré avant le dernier caractère sélectionné.

Il n'est pas possible de faire appel à la sélection directe pour positionner le point d'insertion correcte-

ment à chaque fois:

```
tell application "RagTime 5" -- Génère une erreur
    set insertion point before selection to "-"
    set insertion point after selection to "-"
end tell
```

La solution pour un tel cas de figure est de redemander une nouvelle référence sur la sélection après la première modification:

```
tell application "RagTime 5"
    set mySelection to selection
    set insertion point before mySelection to "-"
    set mySelection to selection
    set insertion point after mySelection to "-"
end tell
```

Évidemment, dans ce cas particulier, le problème pourrait être résolu en changeant l'ordre des instructions dans le premier script:

```
tell application "RagTime 5"
    set mySelection to selection
    set insertion point after mySelection to "-"
    set insertion point before mySelection to "-"
end tell
```

Commencer par la fin pour finir avec le début est une bonne stratégie lorsqu'il s'agit d'insertion mais pas quand il s'agit de suppression. Cela est valable, par exemple, pour ajouter des objets Dessin sur une page.

Quand utiliser la commande "try" avec les sélections

Lorsqu'un script fait appel à la sélection, le principal problème est que vous ne savez jamais ce que l'utilisateur a sélectionné. La plupart de vos scripts ne fonctionneront (et ne seront valables) seulement qu'avec un certain type d'objets. Il y a deux approches possibles: d'abord vérifiez si l'objet est du bon type ou bien essayez mais vérifiez si par la suite l'opération a échoué ou pas.

Si le type des objets est clairement déterminé, la première approche est la meilleure. Si votre script ne marche qu'avec des rectangles, vérifiez si la sélection est bien un rectangle. ("(class of selection) = rectangle" doit renvoyer vraie ("true")).

Beaucoup d'actions ne sont valable que pour une variété d'objets. Un script qui dessine une ligne de guide verticale à travers le milieu d'un objet pourrait être utilisable avec des rectangles, des polygones, des textes graphiques, ... Effectivement, il fonctionnera dans presque tous les cas où un objet a une propriété du type position. Dans de tels cas, utilisez la commande “try”. Créer un script qui tient compte de tous les cas de figure est une tache lourde.

Je suppose un document contenant du texte et une feuille de calcul. Le tableau doit être installé par le script au début du texte sélectionné.

```
tell application "RagTime 5"
    set textSelection to selection
    set myTable to a reference to table "Feuille de calcul 1" of document 1
    if (class of selection) is text then
        set insertion point before textSelection to myTable
    end if
end tell
```

L'expression "Insertion point" ne peut être utilisée que s'il s'agit de texte, alors nous savons de quel type doit être la sélection. L'un des avantages est que vous pouvez générer un message d'erreur clair si la sélection n'est pas bonne:

```
...
    if (class of selection) is text then
```

```
        set insertion point before textSelection to myTable
    else
        display dialog ¬
            "Ce script ne fonctionne que si du texte est sélectionné." buttons {"OK"} default button 1
    end if
...
```

Je recommande toujours de précéder les opérations sensibles par l'instruction "try". Il est impossible de prévoir tous les types de sélection.

Voici le script qui reprend l'exemple de la ligne de guide:

```
tell application "RagTime 5"
    try
        set {X, Y} to position of selection
        make new vertical guide with data X
        make new horizontal guide with data Y
    end try
end tell
```

Si cela échoue, pour une raison ou une autre, rien ne se passera. En général, cela n'est pas acceptable puisqu'il rendra l'utilisateur perplexe. D'un autre côté, puisque votre script ne sait pas ce qui s'est passé, un message d'erreur ne peut être qu'imprécis. Ajoutez le message généré par AppleScript à votre propre message:

```
property errorText1 : "Aucune ligne de guides créés.
Avez-vous vraiment sélectionné un objet Dessin?"
property errorText2 : "Message d'erreur AppleScript: "
tell application "RagTime 5"
    try
        set {X, Y} to position of selection
        make new vertical guide with data X
        make new horizontal guide with data Y
    on error asMessage number errorNumber
        display dialog errorText1 & return & return & errorText2 & ¬
            errorNumber & ", " & asMessage buttons {"OK"} default button 1
    end try
end tell
```

(Définir toutes les chaînes de caractères comme propriétés au début d'un script rend plus simple sa maintenance.)

Divers

L'instruction "path to me"

Si le script est exécuté dans RagTime, "path to me" renvoie le chemin d'accès qui mène au fichier de l'application RagTime. Lorsque que vous concevez un script dans un éditeur de scripts, le résultat n'est pas le même. Si vous voulez retrouver le même résultat que celui obtenu lorsque le script est exécuté dans RagTime, utilisez la propriété "parent".

Durée de vie des propriétés

Lors de l'exécution dans RagTime, les scripts ne se modifient jamais. La seule chose qui change dans un script lors de son exécution dans RagTime est le contenu des variables et des propriétés. Le contenu des propriétés est toujours placé en mémoire vive, jamais sur le disque dur. Le contenu d'une propriété est conservé jusqu'à la fermeture du document si le script a été sauvegardé dans le document, ou bien

jusqu'à la fermeture de RagTime si le script a été sauvegardé dans les accessoires de l'application.

```
property myProperty : 1
display dialog "La valeur de la propriété est: " & myProperty as text buttons {"OK"} default button 1
set myProperty to myProperty + 1
```

À la première exécution, le script affichera le chiffre 1. Si vous l'exécutez à nouveau, le nombre affiché changera pour 2, puis 3, ...

Si le script est enregistré dans le document et si vous fermez ce document, ces modifications seront oubliées. Si vous ouvrez le document à nouveau, le script affichera le chiffre 1 lors de sa première exécution.

Si le script est sauvé dans les accessoires de RagTime, la propriété sera initialisée au chiffre 1, après avoir quitter et relancer RagTime.

Identification des fichiers au format RagTime

Lorsque les fichiers viennent d'une plate-forme Windows ou bien ont été transféré via Internet, il arrive souvent qu'ils n'aient plus les attributs de type (TYPE) et de créateur (CREATOR). Si un document RagTime a une origine Windows, son nom se terminera par l'extension ".rtd" (ou bien ".rtt" pour un modèle). Il y a tout de même une façon efficace pour identifier un tel document sans faire appel à son nom: tout fichier RagTime (et ceci depuis la version 4 de RagTime) a son contenu qui commence toujours avec la même séquence de 12 octets.

Le script suivant vérifie si un fichier est au format RagTime, tout en maîtrisant les risques d'erreurs d'exécution:

```
set theFile to (choose file)

set RTMagicBytes to "C#+D§CM•Hdr◊"
set magicBytes to ""
try
    set fileToTest to (open for access theFile)
    set magicBytes to (read fileToTest to 12)
    close access fileToTest
on error
    try
        close access fileToTest
    end try
end try
set isRTFile to (magicBytes = RTMagicBytes)
```

Commande "finish calculation"

Votre script peut entrer des valeurs dans une feuille de calcul et par la suite accéder aux valeurs résultant de calculs effectués par le tableau. Avec une application multiprocessus, il est impossible de savoir si les calculs ont été effectivement accompli lorsque le script désire accéder aux résultats. Utilisez la commande "finish calculation" pour synchroniser la tâche (thread) issue de l'exécution du script avec celle générée par la feuille de calcul.

Notez que la chronologie des tâches n'est pas la même si le script est testé dans un éditeur de scripts externe ou bien si il est finalement exécuté dans RagTime. Les problèmes ne surviennent que si la commande “finish calculation” est omise dans un script qui s'exécute dans RagTime.