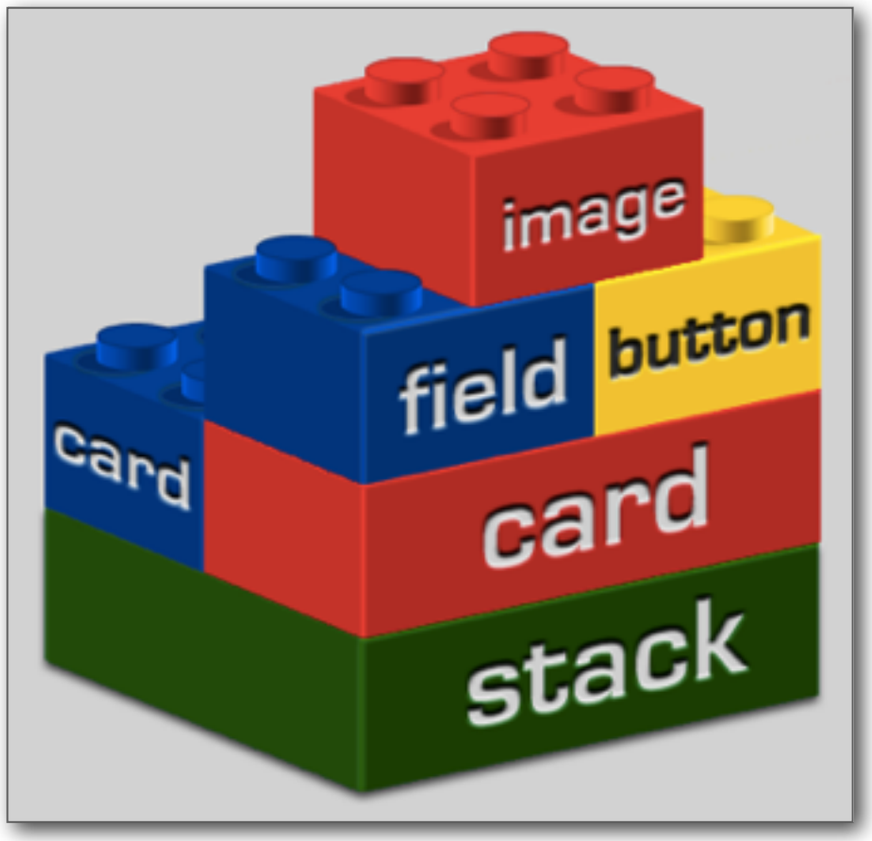
LIVECODE

Community Edition



DÉVELOPPEMENT LOGICIEL MULTI-PLATEFORME
WINDOWS / MAC / LINUX / ANDROID / IOS

UNE PRÉSENTATION DE JEAN-MARC TEULÉ

LiveCode, Why I Use It

Est-ce un kit de développement logiciel (SDK) ? Oui.
Est-ce un langage de programmation ? Oui.
Est-ce un outil de conception d'interface ? Oui.
S'agit-il d'un système de prototypage rapide ? Oui.

Qu'est-ce que n'est pas, alors, LiveCode ?

Je ne sais pas.

Je trouve qu'il est très difficile d'expliquer aux gens ce qu'est LiveCode.

Javascript est un langage de script Web côté client.
Très bien.
PHP est un outil de script web côté serveur.
Très bien.
Raspberry Pi est une carte micro-ordinateur basé sur Linux.
Très bien.

LiveCode est ... Beaucoup de choses, plus difficiles à transmettre que de dire «couteau suisse», ce qui serait une comparaison tentante mais très mauvaise. Je ne peux pas le dire en une phrase, pas même dans une page sur le web, mais je vais essayer ici.

Vous pouvez faire de réels programmes d'application avec LiveCode, et ils fonctionnent comme n'importe quelle autre application réelle sur votre bureau, votre smartphone, votre tablette.

Robert Cailliau

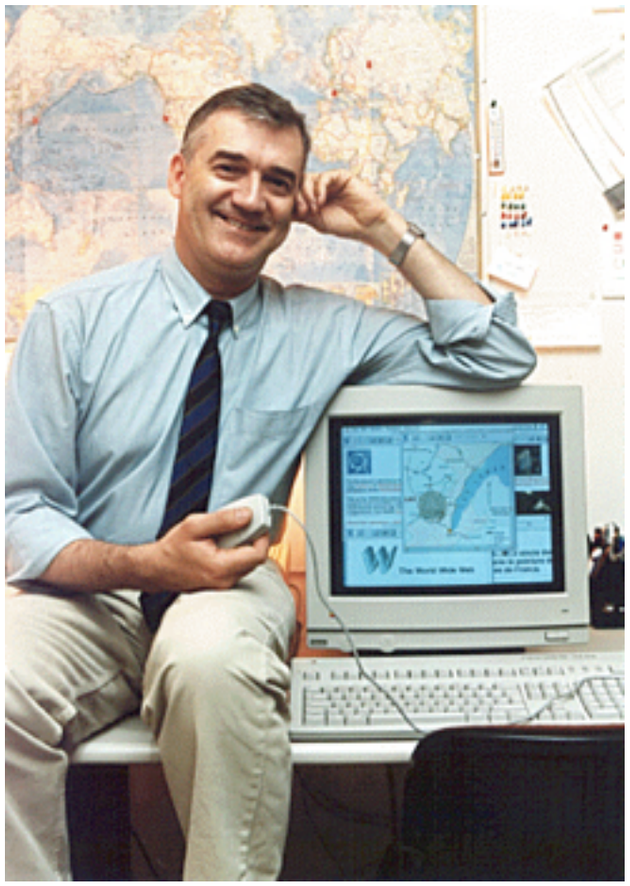


Photo CERN

Robert Cailliau a été le premier collaborateur de Tim Berners-Lee pour la création du projet du **World Wide Web**.
Avocat infatigable du Web, il mit en place les conférences sur le **World Wide Web** et resta membre du comité d'organisation de **1994** à **2004**.

Qu'est-ce donc que LiveCode ?

LiveCode est un **IDE** (**I**ntegrated **D**evelopment **E**nvironment) produit par la société Ecossaise **RunRev** (Runtime Revolution) qui se présente sous la forme d'un **RAD** (Rapid Application Development) dont la finalité est de créer soi-même, aisément, des applications multi-plateforme (pour ordinateurs Windows, Linux, Mac et mobiles Android, iOS).

C'est un produit commercial, mais depuis ce printemps 2013, RunRev a décidé de le faire évoluer et de proposer en plus une version complète, libre et Open Source, sous licence GPLv3.

Pour mener à bien cette transition, la société a lancé un appel à financement sur Kickstarter avec un objectif à 350 000 £ et a obtenu 500 000 £ !

Vu le succès de la démarche, **RunRev** a décidé de distribuer immédiatement et gratuitement la version actuelle sous licence **Open Source**, dite **LiveCode Community**.

linuxfr.org : Livecode est libéré

developpez.com : LiveCode devient open source

Il existe aussi une version **serveur** de **LiveCode**, disponible également sous licence **GPL3** . Le moteur de la version serveur est différent, et possède une syntaxe adaptée pour le rendre approprié à une utilisation en ligne de commande en tant que processeur CGI.

Cela lui permet donc de traiter des scripts à partir de fichiers texte. On peut donc mélanger du script **LiveCode** avec du **HTML**, du **CSS** et du **JavaScript** comme avec d'autres langages de script Web. Des exemples ici.

Mais ceci n'est pas l'objet de cette présentation. Nous resterons sur une description de la version client, son histoire, son utilisation et comment apprendre son langage.

L'intérêt tout particulier de **LiveCode** est de permettre à des non professionnels du développement de concevoir eux-même des applications, même sans connaissance préalable d'un langage de programmation de type classique.

Le langage qu'il utilise pour l'écriture des scripts a la particularité d'être proche de l'anglais courant. C'est bien pour les anglophones, un peu moins bien pour les autres, mais il reste tout de même très accessible et surtout très lisible même

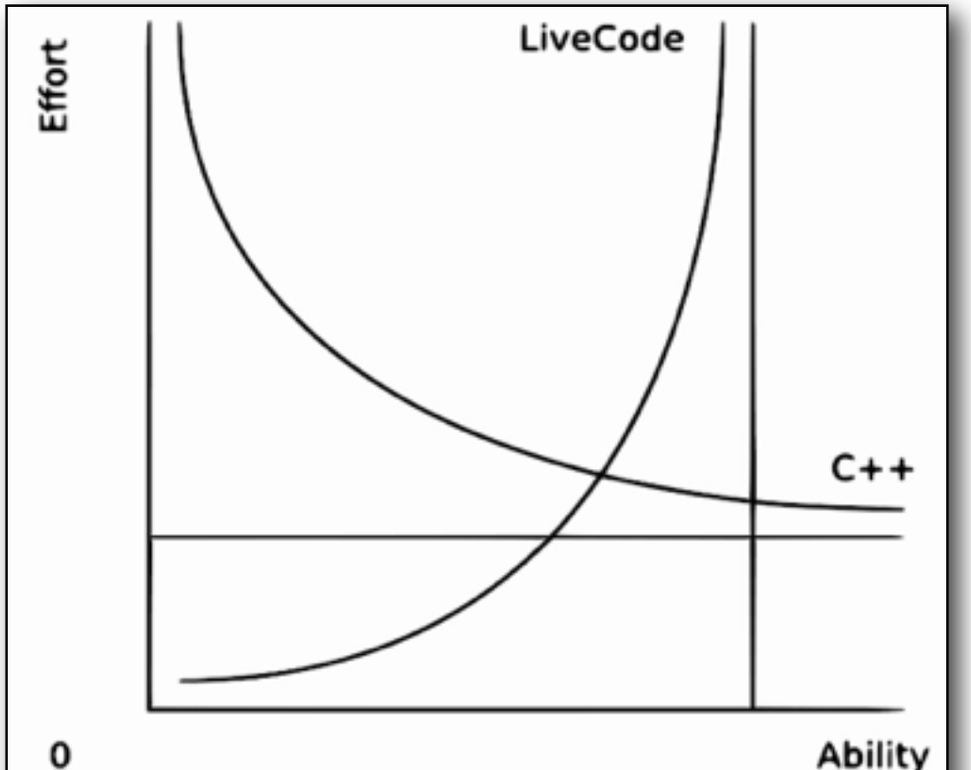
pour un profane ou pour une relecture ultérieure.

C'est pour cela que ce type de langage, très proche de l'humain, est dit de 4ème génération (tout comme le **SQL** des bases de données).

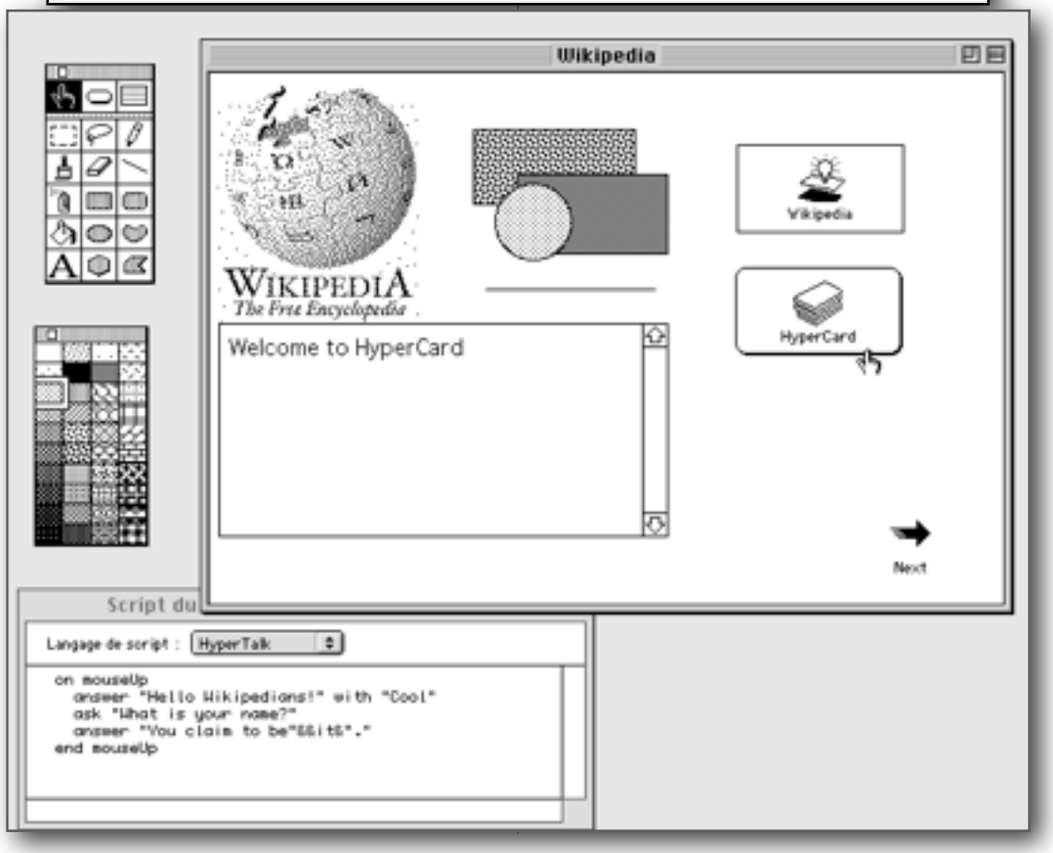
Par opposition, nous trouvons du code de bas niveau, proche de la machine, dit de 2ème génération comme l'assembleur, difficile à maitriser et non portable (lié au type de processeur).

Entre les deux, les langages dits de 3ème génération sont les plus connus (**C**, **C++**, **Java**, etc...) mais ils demandent quand même une réelle maîtrise préalable.

On voit sur l'illustration centrale du dessus (source) que la courbe d'apprentissage de **LiveCode** démarre très bas mais l'effort augmente fortement si on cherche les limites du système d'exploitation, tandis qu'avec un langage de type **C++** l'effort sera maximum dès le départ pour ensuite diminuer rapidement.



Compromis entre effort et capacité



Le langage utilisé par **LiveCode** est une sur-couche, car le noyau de **LiveCode** fonctionne en **C++**.

Ce langage fait partie de la famille des **xTalks**. Le premier de cette famille fut **HyperTalk**, créé par **Dan Winkler**. Il motorisait le célèbre **HyperCard** de **Bill Atkinson** sorti en **1987**.

Celui-ci popularisa l'utilisation des liens de type **hypertexte** en permettant de relier des informations (mots, textes ou objets) placées sur des cartes (**cards**), elles-mêmes organisées en piles (**stacks**), pour créer de véritables applications, ou des bases de données personnelles.

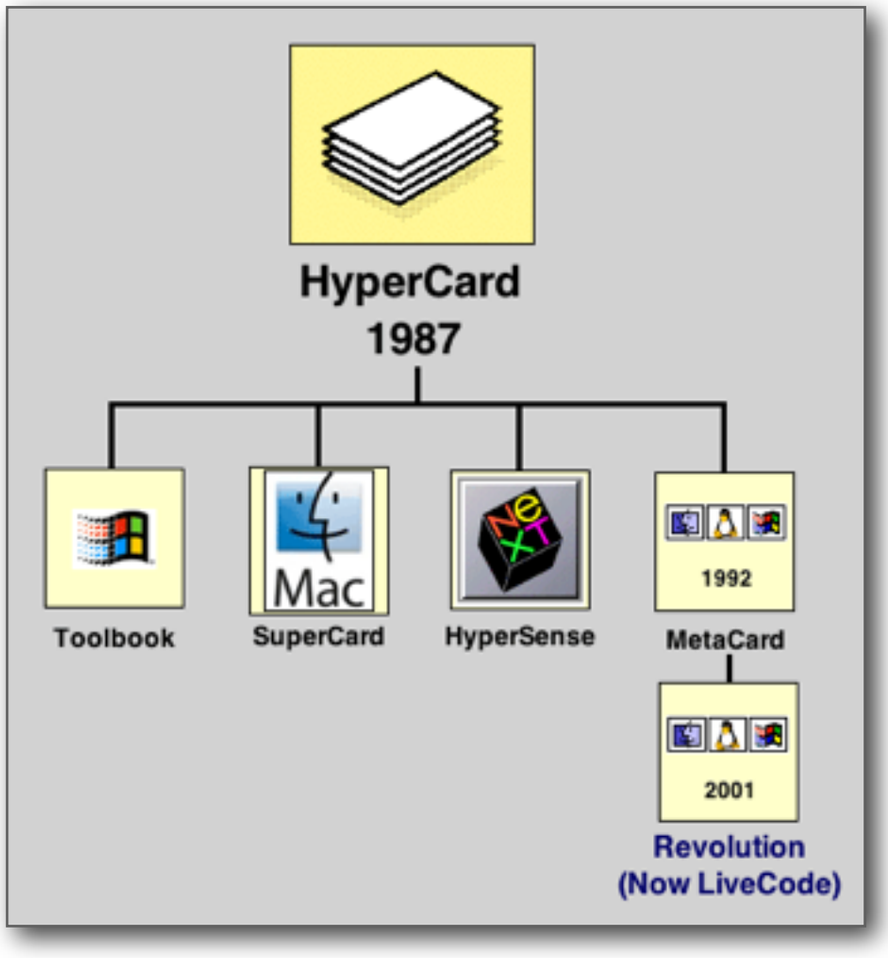
Tout ceci bien avant la création du **World Wide Web** (**1989/90**) et du premier navigateur **Web** par **Tim Berners-Lee**, aidé de son collaborateur **Robert Cailliau**. Ils reprirent le concept d'**hypertexte** pour l'étendre sur plusieurs machines distantes via Internet.

Bill Atkinson, un peu plus tard,

déplora que, s'il avait seulement réalisé la puissance des piles orientées réseau, au lieu de se concentrer sur des piles locales sur une seule machine, **HyperCard** aurait pu devenir le premier navigateur **Web** de l'histoire.

HyperCard cessa définitivement d'être distribué en **2004**.

Mais le langage **HyperTalk** eut une descendance, en particulier **MetaTalk** moteur du logiciel **MetaCard** en 1992.



Il améliora le concept initial d'**HyperCard** et devint **multi-plateforme**. Puis advint **Revolution** de **RunRev** (société basée à **Edimbourg**). D'abord conçu en **2001** comme IDE expert pour **MetaCard**, il absorba complètement le langage **MetaTalk** alors renommé **Transcript** lorsque **RunRev** fit l'acquisition de **Metacard** en **2003**. Puis en **2010**, **Revolution** changea de nom pour s'appeler **LiveCode**.

Voici pour ce court historique des racines de **LiveCode** qui nous a fait remonter au début de l'informatique personnelle.

Aujourd'hui, la création de la version **Open Source**, appelée **LiveCode Community**, autorise non seulement son utilisation par quiconque mais également la libre distribution des logiciels produits avec (sous réserve du respect de la licence virale **GPL3**).

L'éditeur de **LiveCode** vante une grande facilité d'utilisation et d'assimilation, mais ce dernier n'en reste pas moins un outil de développement logiciel et à ce titre il faut prendre le temps d'apprendre, d'errer, de se tromper, de reprendre.

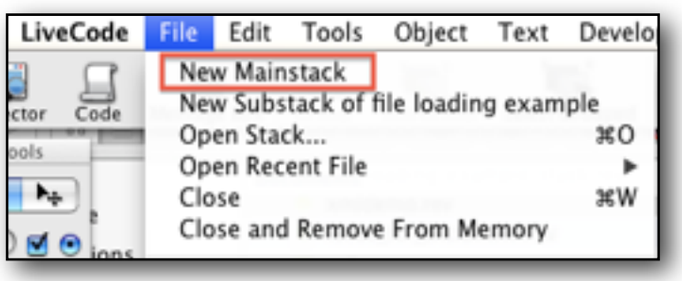
Mais **LiveCode** a une qualité supplémentaire à la lisibilité de son code, celle, justement, de faciliter l'apprentissage par essai/erreur. **LiveCode** permet d'un clic de passer de l'édition à l'exécution, sans phase de compilation intermédiaire fastidieuse et longue.

Pour comprendre ceci, nous allons détailler un peu son principe de fonctionnement au travers d'un exemple extrêmement sommaire.

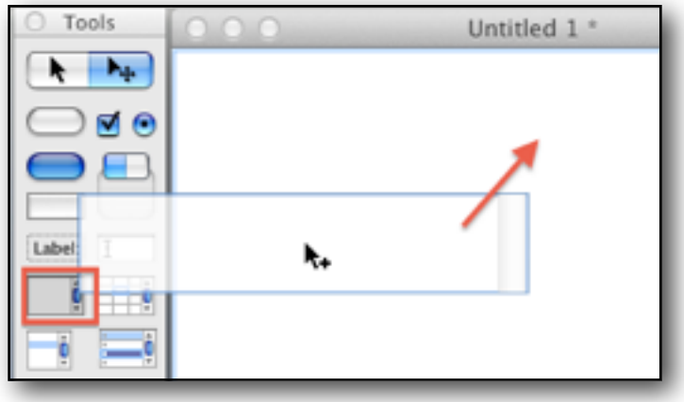
Comment utiliser LiveCode ?

LiveCode est d'abord un constructeur d'interface et pas seulement un langage. Il propose via son IDE, comme d'autres outils du même genre, une palette d'outils contenant des objets qui sont soit des contrôles (boutons, sélecteurs, tirettes) soit des contenants (champs, tables, grilles de données). Le langage de **LiveCode** se chargera de gérer les événements entre les éléments actifs de votre interface.

Prenons d'une interface ultra basique se composant de seulement 3 éléments : 1 fenêtre contenant un champ texte et un bouton. Pour créer la fenêtre on choisira dans le menu l'item ***New Mainstack*** (Pile principale).

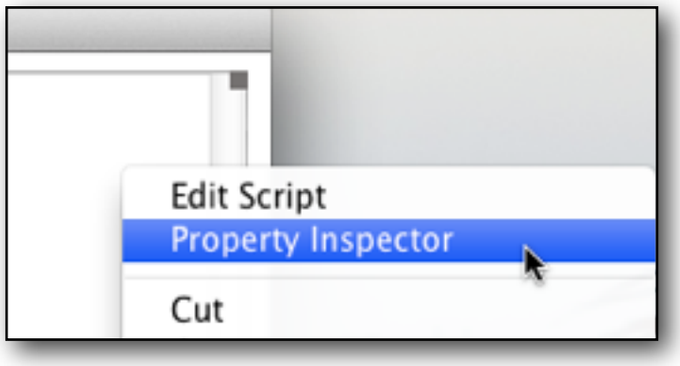


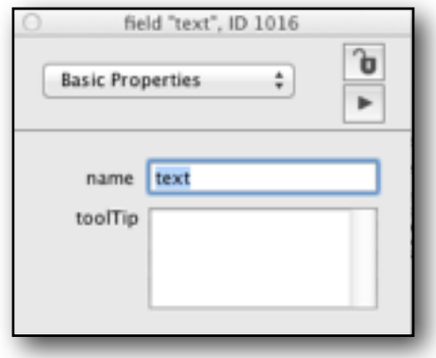
S'affiche alors une fenêtre vide dans laquelle il nous reste à glisser l'objet champ texte puis un objet bouton.

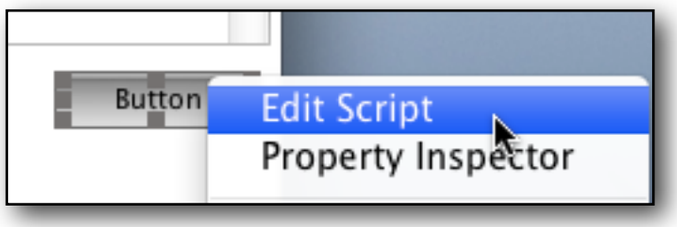


Il suffit de positionner et de dimensionner ces éléments à sa guise.

Ensuite faire un clic droit sur le champ texte pour ouvrir ***Property Inspector*** (le gestionnaire des propriétés des objets de **LiveCode**). Dans le champ ***name*** indiquez le nom de votre champ, par exemple text pour faire court. Refermez ***Property Inspector***. Vous pouvez faire de même sur chaque objet de votre pile.

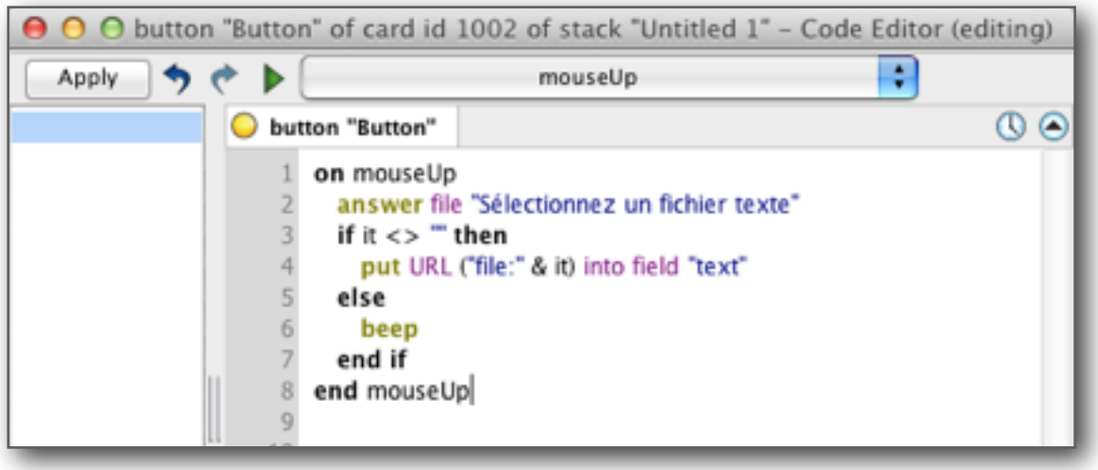






Faites un clic droit sur le bouton pour ouvrir l'éditeur de script.

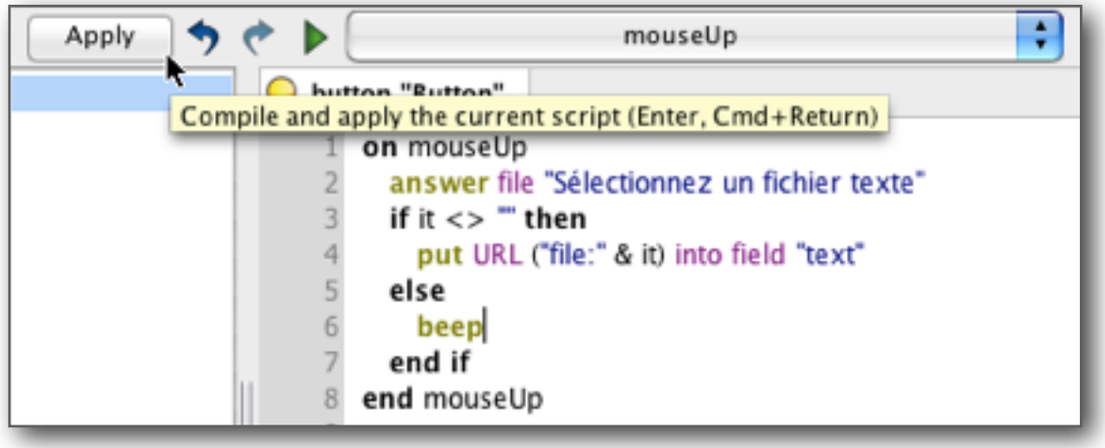
On y écrit le code qui fera fonctionner le bouton, c'est à dire sélectionner un fichier texte pour copier son contenu dans le champ créé.



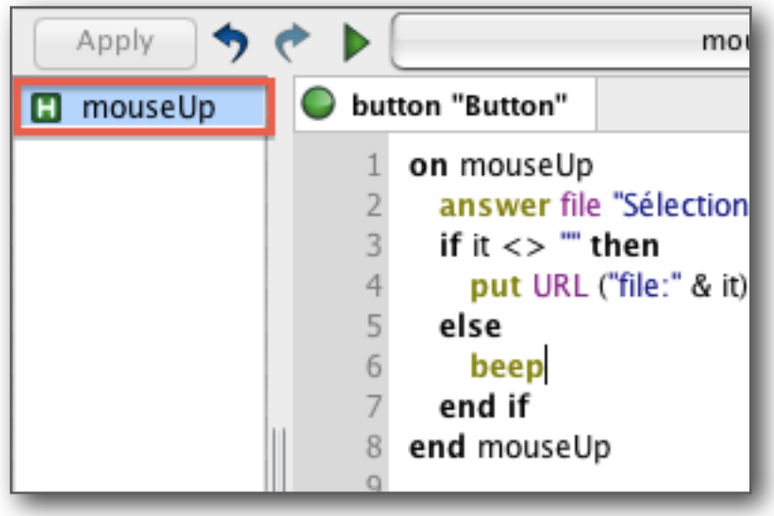
Le code est contenu entre **on mouseUp** et **end mouseUp**. L'événement **mouseUp** correspond au moment ou le doigt de l'utilisateur relâche la bouton de la souris après avoir cliqué sur un objet, ici le bouton. Cet **événement** est capturé par le gestionnaire appelé **on mouseUp** qui appliquera le code prévu : récupérer le contenu du fichier sélectionné et l'écrire dans le champ texte.

- ***answer file*** : est une commande **LiveCode** qui ouvre le sélecteur de fichier de votre système d'exploitation avec en titre la phrase entre guillemets.
- ***it*** : est une variable automatique éphémère qui contient le résultat du choix dans les boites de dialogues.
- ***put URL*** : **URL** est un conteneur spécial qui gère les adresses de type **file** (fichier), **ftp**, **http**. Dans ce cas on copie le contenu du fichier (*put URL... into field...*) à l'adresse récupérée dans la variable ***it*** dans le champ nommé text
- ***beep*** : est une commande pour indiquer à votre système d'émettre un bip d'erreur (ou de validation).

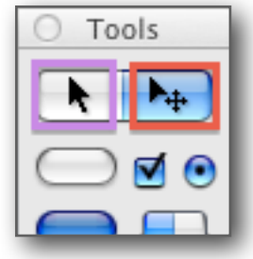
Un fois le code écrit il faut le valider pour le tester. Appuyer sur le bouton ***Apply***, une sorte de compilateur instantané.



Si une erreur de syntaxe est détectée, la validation s'arrête pour indiquer approximativement l'endroit de l'erreur. Si aucune erreur n'est détectée, le raccourci du gestionnaire (***handler*** d'où la lettre **H**) est inscrit dans la colonne de gauche de l'éditeur.

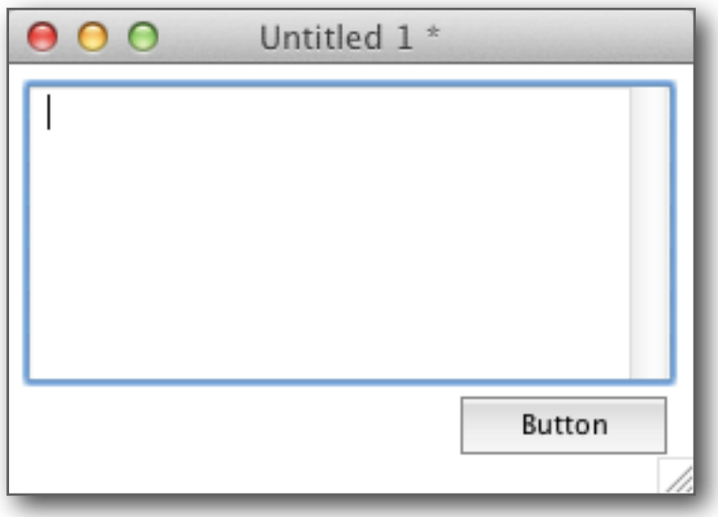


Pour tester le code final, et vérifier qu'il produit bien l'effet escompté, il suffit de basculer en mode exécution. La palette d'outil **LiveCode** possède en son sommet un bouton bascule permettant de passer instantanément de l'état d'édition (bordure rouge) à l'état d'exécution (bordure mauve) et inversement.



Pour pouvoir créer, éditer les objets, pour y insérer ou modifier un script, il est nécessaire de les désactiver du flux d'événements. Ils deviennent alors insensibles aux événements auxquels ils sont censés réagir et peuvent être manipulés tranquillement.

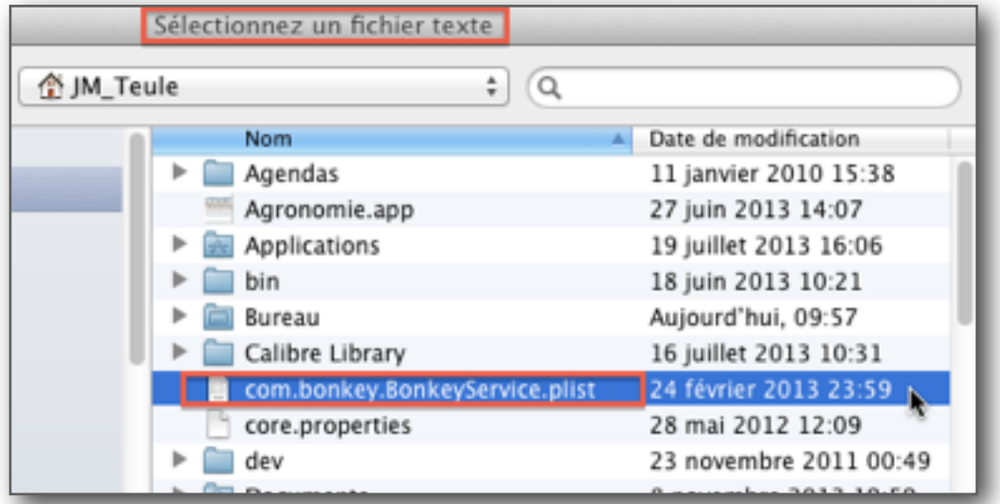
Une fois les modifications apportées, il suffit de basculer sur le mode exécution pour tester immédiatement le fonctionnement de l'élément édité en même temps que l'ensemble de l'application en construction.

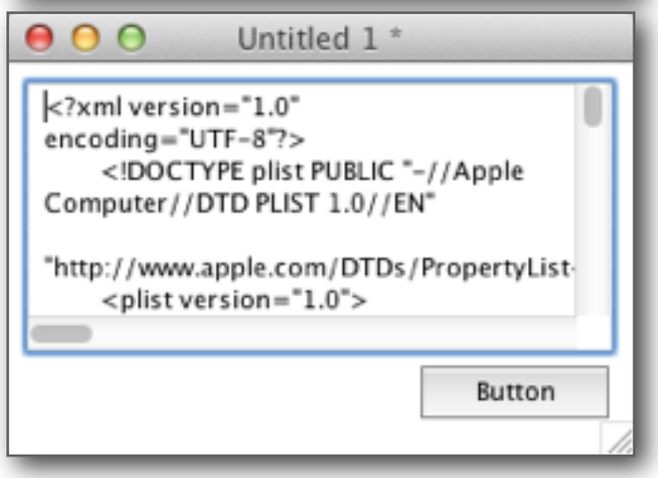


Notre interface rudimentaire est maintenant en mode exécution, n'attendant plus qu'un clic sur le bouton ***Button***.

Si vous cliquez lentement sur votre bouton droit de souris, vous constatez bien que le relâchement de celui-ci par votre doigt déclenche l'apparition du sélecteur de fichier de votre système à partir duquel vous pouvez sélectionner un fichier texte de votre choix, n'importe où dans l'arborescence de votre système d'exploitation.

Le sélecteur de fichier s'est bien ouvert avec sa barre titre portant le libellé prévu dans la code. Il suffit de faire un double clic sur un fichier texte, par exemple, ici, un fichier de configuration d'application **.plist** (de type **xml**) pour l'afficher dans le champ text, prévu à cet effet.





On peut constater l'apparition d'une barre de défilement verticale, mais pas horizontale.

Il suffit de repasser en mode édition et faire un clic droit sur le champ texte pour ouvrir ***Property Inspector*** et valider la propriété ***hScrollbar***. Aussitôt fait, la barre de défilement horizontale apparaît.

On basculera ainsi autant de fois que nécessaire entre les modes édition et exécution pour modifier son interface et/ou son code. Une fois satisfait, il suffit de transformer votre pile en une application autonome pour votre système, avec les options utiles (non détaillé ici)

LiveCode propose **3 IDE** distincts. Un pour chaque système d'exploitation supporté, à savoir **Windows**, **Mac** et **Linux**.

Actuellement **LiveCode** est en train d'évoluer assez radicalement.

Pour résumer on peut considérer les versions **6.x.x** et inférieures comme les versions **classiques historiques**, les versions **7.x.x** comme transitoire, avec un passage au tout **Unicode** puis les versions **8.x.x** qui sont le nouvel objectif non encore abouti de la modernisation de l'IDE (pas de version stable à ce jour) avec une évolution notable de l'interface, ainsi que l'ajout d'un nouveau type de script (**LCB**) pour des appels à du code étranger, et l'export des applications **LiveCode** en **Javascript** pour les rendre exécutables dans un navigateur.

Privilégiez donc les dernières versions stables classiques (6.x.x) ou Unicode (7.x.x) pour plus de tranquillité.

Chaque version s'installe séparément et n'écrase pas la version précédente ce qui pérennise la maintenance de vos développements.

Cliquez ici pour trouver la page de téléchargement de toutes les versions.

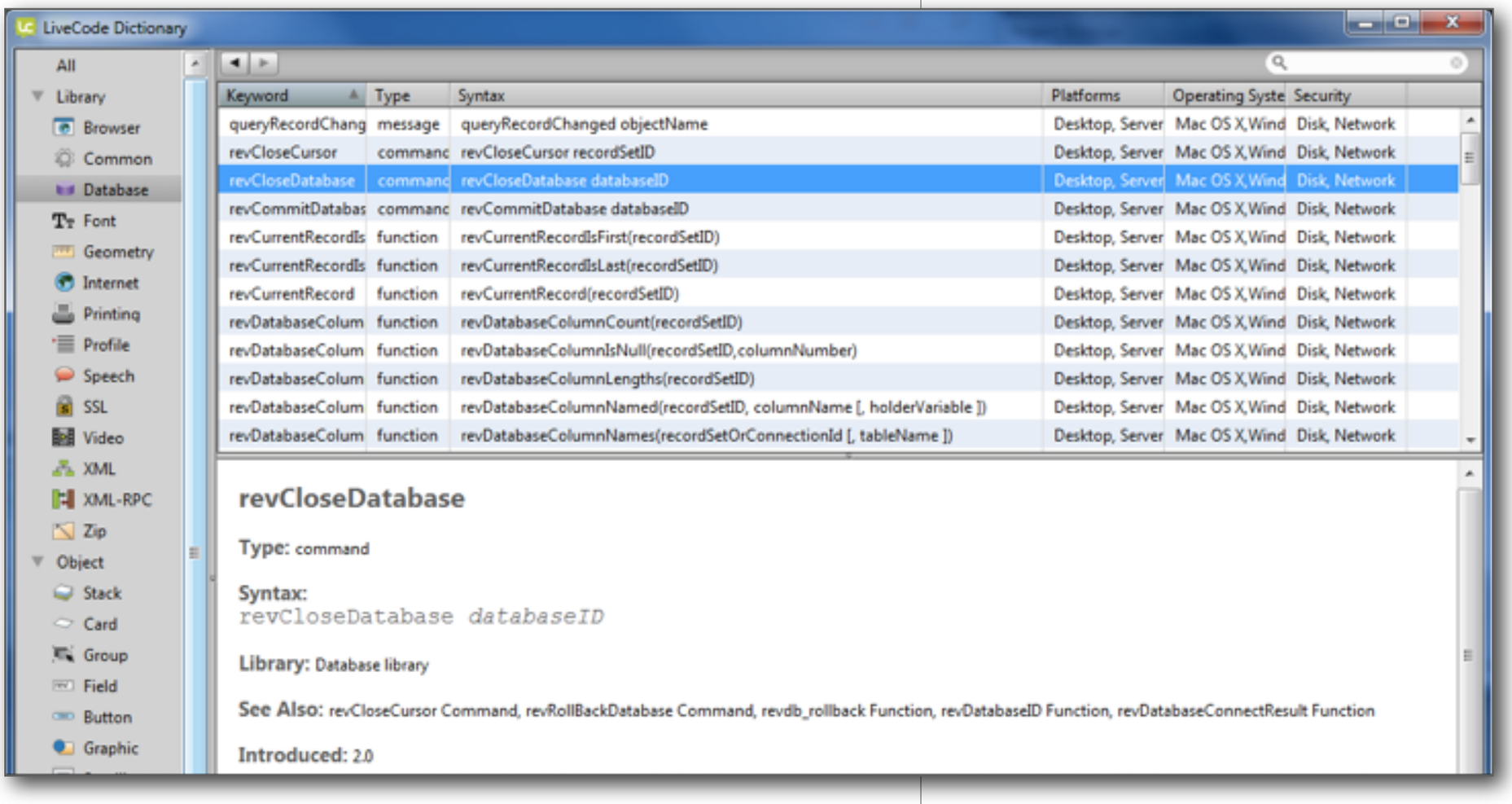
Si vous devez développer une application compatible sur plus d'un système, il vous faudra utiliser l'IDE adapté à chacun des systèmes pour pouvoir valider votre code dans chacun d'eux.

Vous pourrez, alors, soit créer un code unique mais transportable (d'un système à l'autre), soit créer une application par système à supporter. Cette dernière option sera plus simple à développer car moins de boucles de condition à prévoir.

Si votre code interagit peu avec le système hôte et se contente d'ouvrir des sélecteurs de fichiers et des boites de dialogues pour traiter des données, il y aura peu ou pas d'adaptation à faire selon les différents systèmes hôtes de votre future application. Par contre si vous devez faire appel a des fonctions plus spécifiques, par exemple à des instructions en ligne de commande, ou à des chemins de fichiers qui ne sont pas ceux prévus par défaut, il faudra adapter votre code en fonction de l'environnement voulu.

Reste le point le plus délicat : comment apprendre ce langage.

Comment apprendre le langage LiveCode ?



Le langage utilisé par **LiveCode** est constitué d'une syntaxe riche de près de 2500 commandes, fonctions, mots clés, propriétés, opérateurs, messages, objets. Pour retrouver ces éléments et les informations associées, **LiveCode** intègre un dictionnaire doté d'un moteur de recherche et d'une classification par librairie, objet, langage ou tout à la fois.

Pour chaque clé du dictionnaire sont indiqués : le type, la syntaxe, la plateforme supportée et enfin une description détaillée souvent accompagnée d'un ou plusieurs exemple de code. Parfois un commentaire d'utilisateurs complètent la fiche.

L'usage du dictionnaire est indispensable mais non suffisant.

Comment trouver la définition d'un mot clé ou d'une expression que l'on ne connait pas encore ?

Le reproche le plus courant est la relative ambiguité émanant de ce type de langage, à la fois proche et pas toujours semblable.

Les développeurs habitués à d'autres systèmes plus structurés peuvent se retrouver décontenancés par l'approche spécifique de **LiveCode**.

Pour pouvoir appréhender cette logique particulière, une documentation **pdf** est jointe à la distribution de **LiveCode** que vous installez.

Elle est volumineuse (autour de 250 pages), touffue et peu pratique.

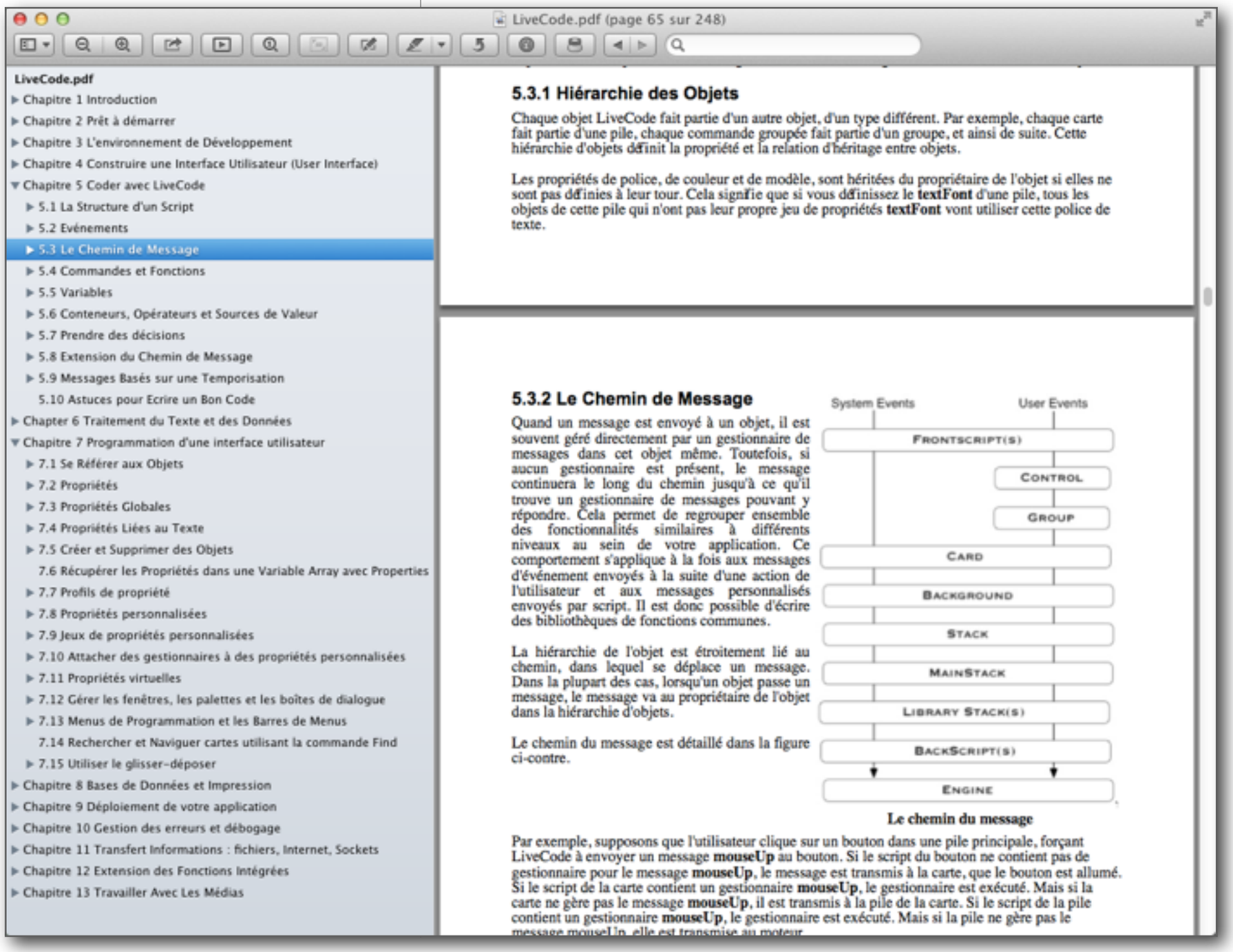
Totalement écrite en anglais elle n'est plus toute récente. A ce jour c'est une version qui date du mois de Novembre 2010 qui est fournie.

Elle est donc incomplète par rapport aux versions actuelles de **LiveCode** mais totalement indispensable pour comprendre le fonctionnement, l'esprit, l'ADN de **LiveCode** et son langage.

J'ai donc procédé à une traduction quasi complète de ce document. C'était en outre l'occasion d'essayer de le rendre un petit peu plus convivial en y apportant la même coloration syntaxique du code que celle qui est proposée dans l'éditeur de code de **LiveCode**.

J'ai rajouté aussi une table des matières intégrant tous les titres jusqu'au niveau 3 pour faciliter la navigation dans l'ensemble du document.

Pensez à la rendre visible dans votre lecteur **pdf** pour en profiter et faciliter la navigation !



Cette source documentaire n'est le seul élément disponible.

L'IDE de LiveCode propose une section comprenant :

- ***User Sample*** : Ouvre un navigateur dans l'IDE pour parcourir des exemples. Très fouillis et un peu fastidieux à parcourir. On retrouve certains éléments également dans les sections suivantes (***Tutorials*** et ***Resources***).

- ***Tutorials*** : ouvre une page web d'un ensemble de cours (*LiveCode Lessons*) pour démarrer en douceur. En anglais et un peu décousu dans la suite des pages, complète la documentation.

- ***Resources*** : ouvre une liste d'exemples divers, de petites démos, de piles ou de code à charger sur divers sujets, certains se croisant avec d'autres de la page ***Tutorials***.

Certaines des piles d'exemples sont de véritables petits cours interactifs dédiés à un sujet particulier :

Par exemple, la pile *SQLite Sampler.rev* fournie dans le dossier *Examples* du dossier d'installation de **LiveCode**.

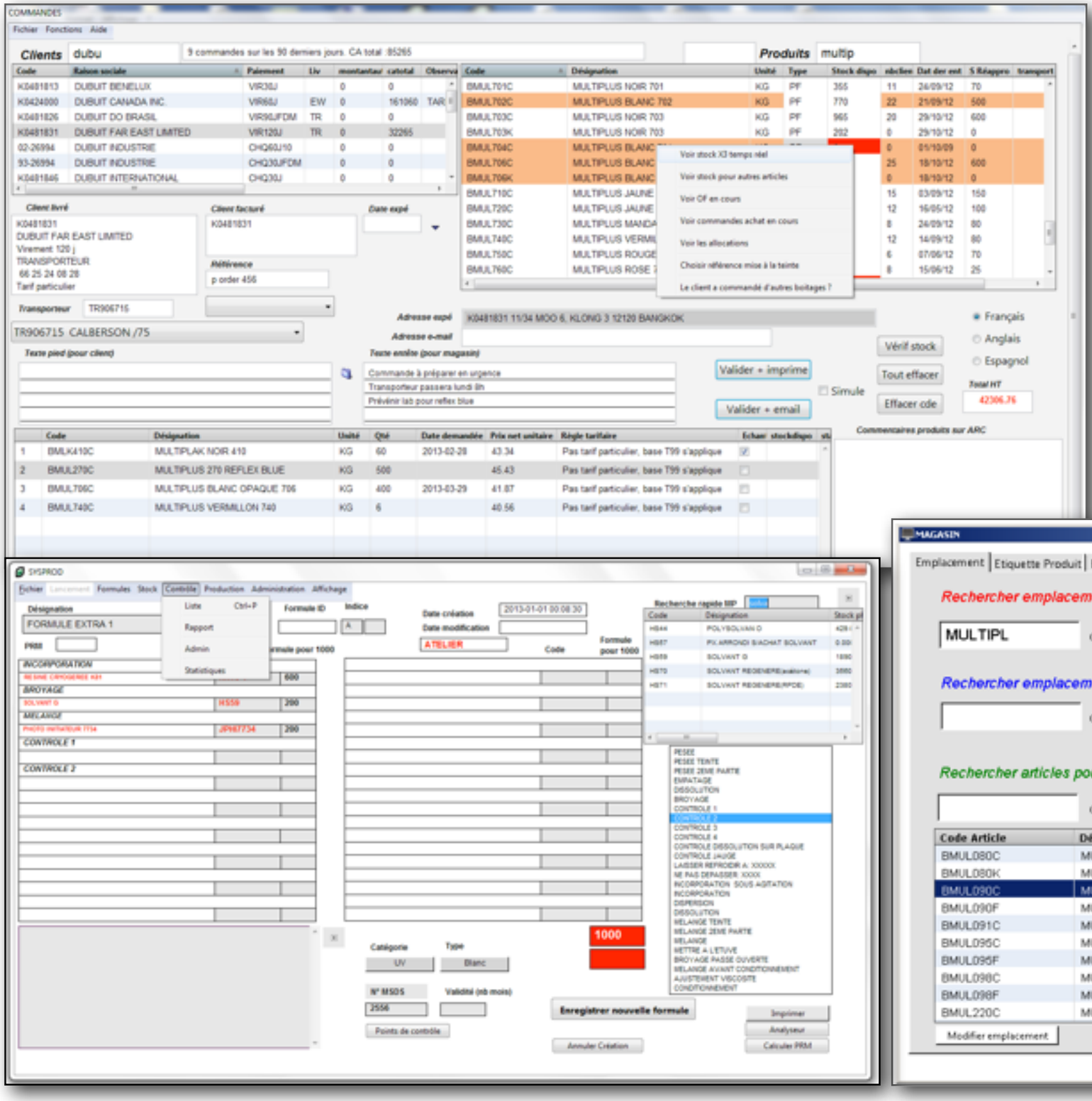
Pour clore ce thème de l'apprentissage il existe quelques forums (Anglais) ainsi que quelques sites tiers (dont certains proposent des librairies, souvent payantes, pour **LiveCode**).

Un développeur a eu la bonne idée d'ouvrir un site Web regroupant toutes les informations et forums sur **LiveCode :** Le site LiveCode SuperSite de Scott McDonald.

Un lien intéressant pour débuter : Topics in LiveCode Programming de Brigham Young University

Malheureusement, rien d'intéressant en Français, pour l'instant.

LiveCode, quelques exemples d'applications



Le fabricant français, **Encre Dubuit**, dont le cœur de métier est la production d'encre, est géré via un logiciel **LiveCode** interface de leur système ERP (pour les mouvements de stocks par exemple) et lié à un serveur **MySQL** local.

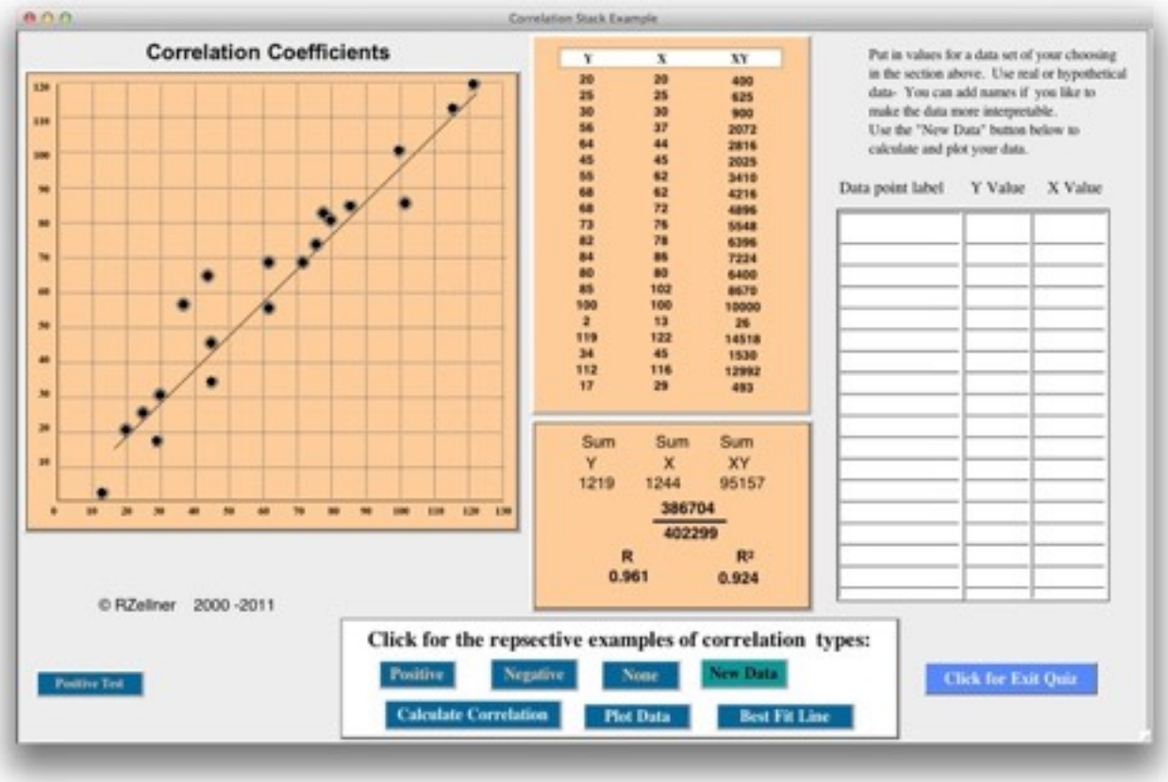
Même leur site Web fonctionne sur la base de **LiveCode Server** :

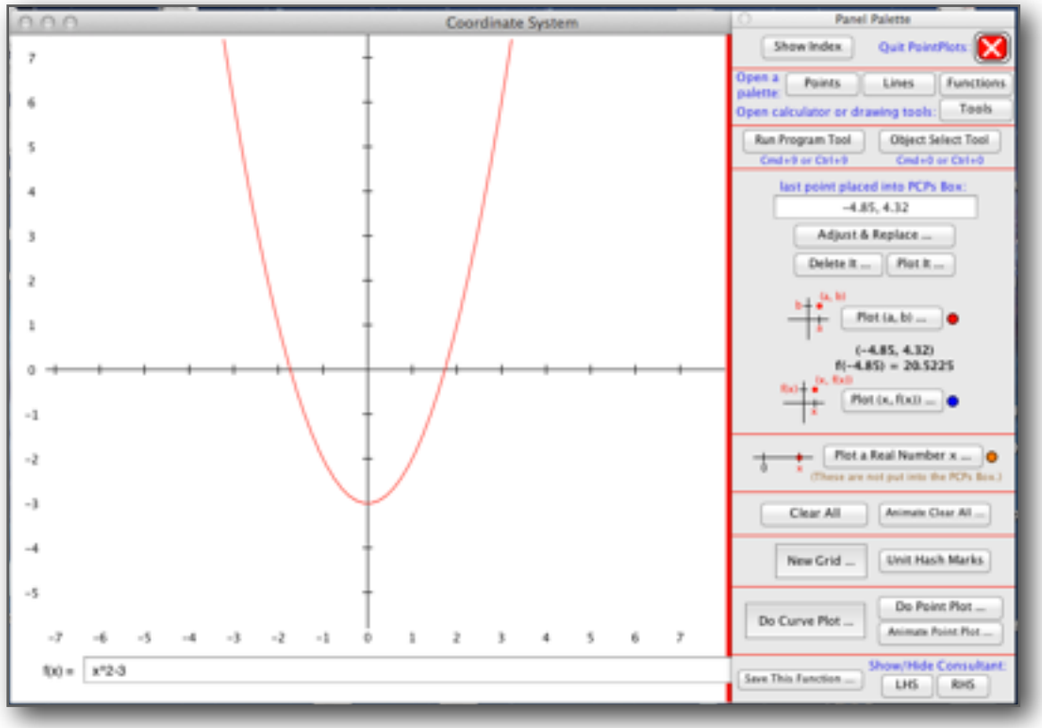
LiveCode, quelques exemples d'applications

Ronald Zeller a construit cette application interactive en LiveCode afin que ses élèves puissent explorer les coefficients de corrélation. Les étudiants peuvent faire glisser les points de données vers de nouveaux emplacements sur la carte et puis recalculer les valeurs d'observer l'effet.

Ils peuvent également créer un nouvel ensemble de données, puis calculer et tracer ces données pour explorer plus avant les principes.

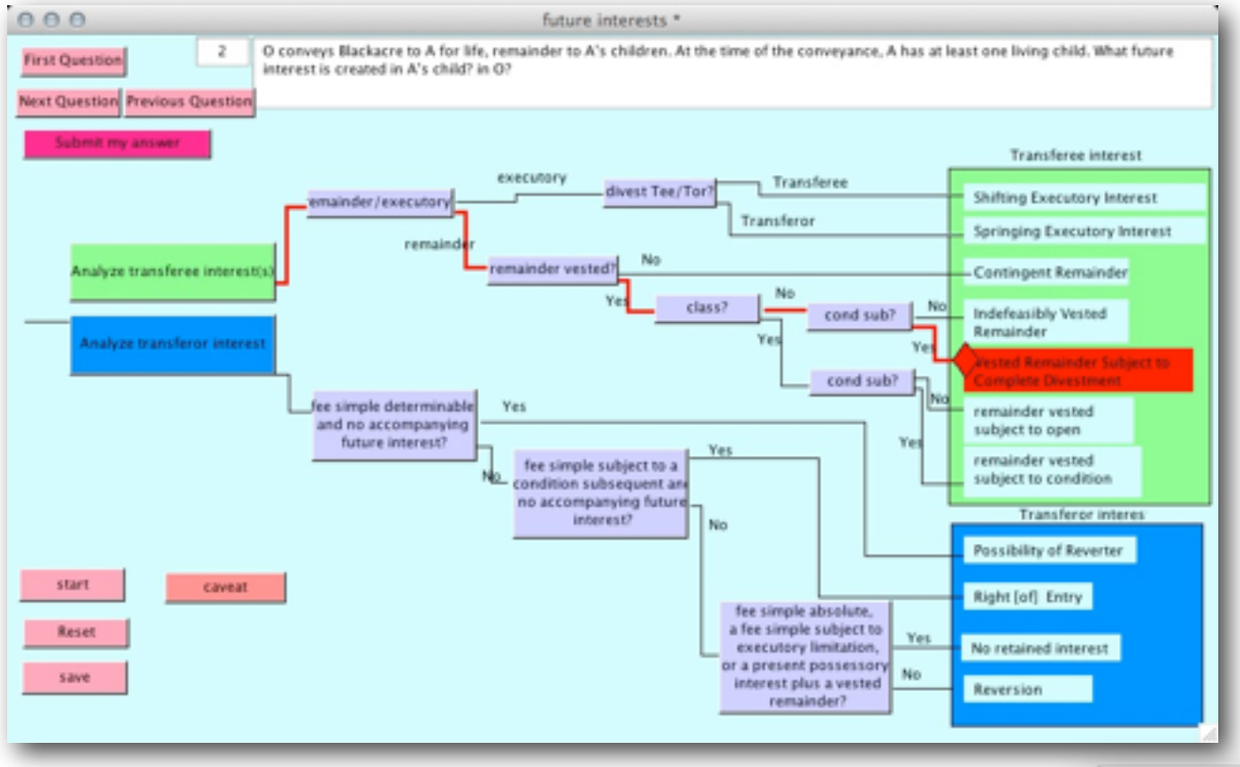
Remarque : Les points du tracé paraissent plus grands que la normale afin de permettre la sélection et le déplacement au doigt sur un iPad.





A. N. DiVito, Ph.D, a développé **PointPlots** dans **LiveCode**. Il s'agit d'un programme pédagogique pour l'enseignement (polygones, les relations, les fonctions, la théorie de la fonction inverse, en coordonnées polaires, équations paramétriques, droites sécantes et tangentes, série de Taylor, arbitraires et régulière sommes de Riemann, etc .).

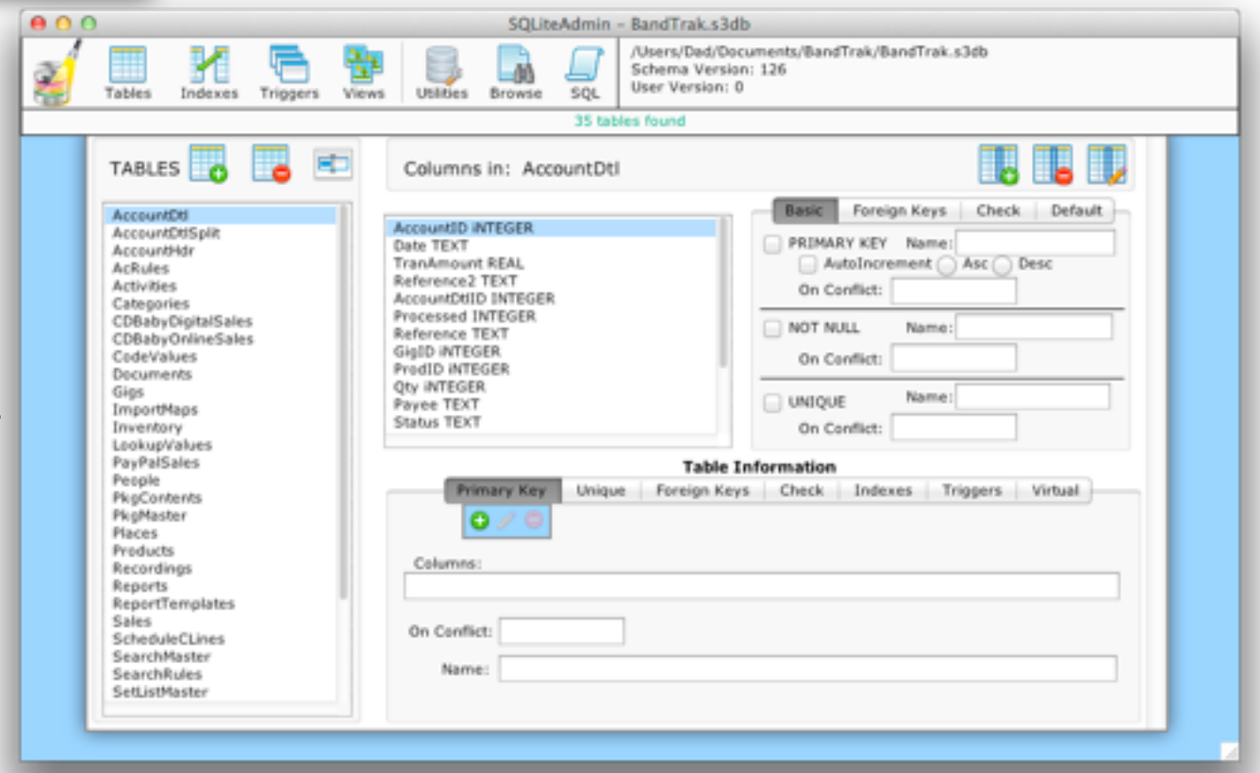
LiveCode, quelques exemples d'applications



David Johnson a utilisé **LiveCode** pour développer des jeux d'apprentissage juridiques à l'usage des étudiants en droit de la **New York Law School**.

Administrer les bases de données **SQLite**

Peter Haworth a développé une application d'administration de base de données **SQLite** entièrement développée avec LiveCode.



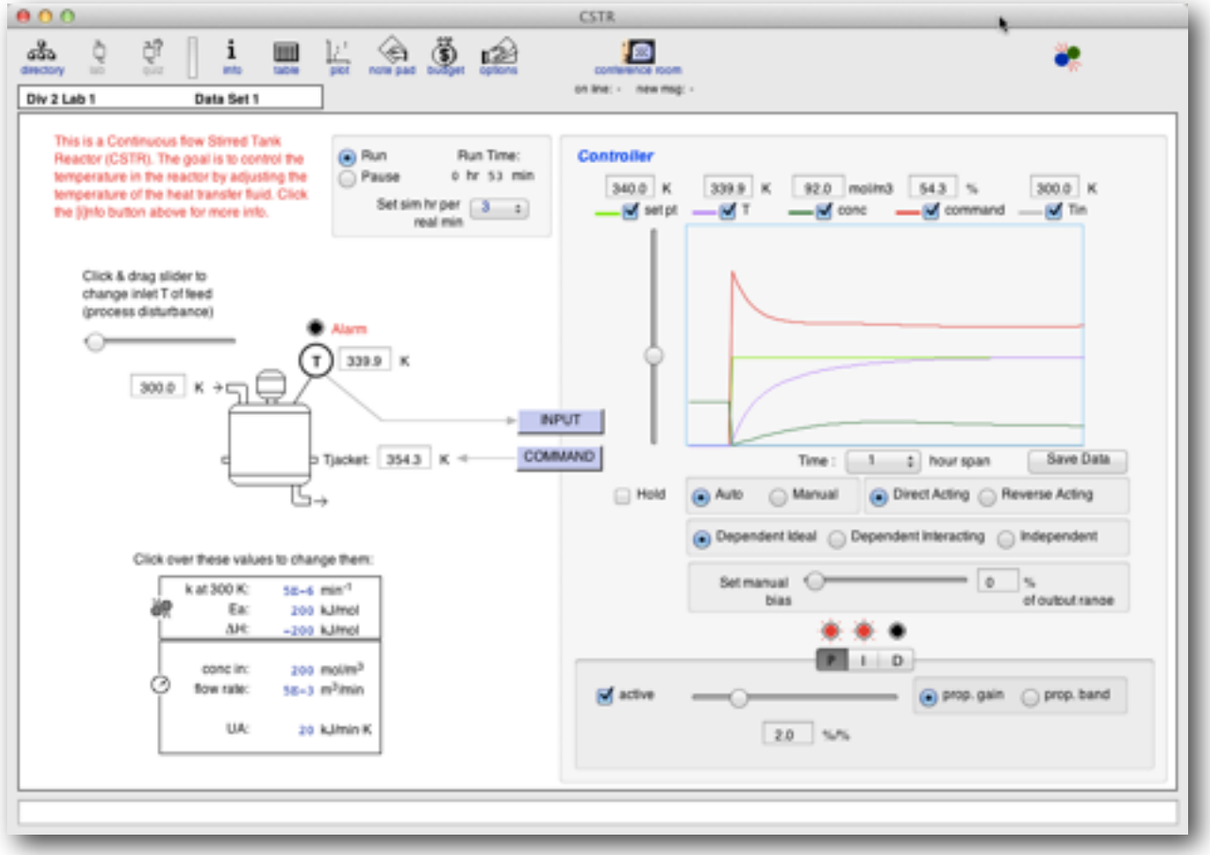
LiveCode, quelques exemples d'applications

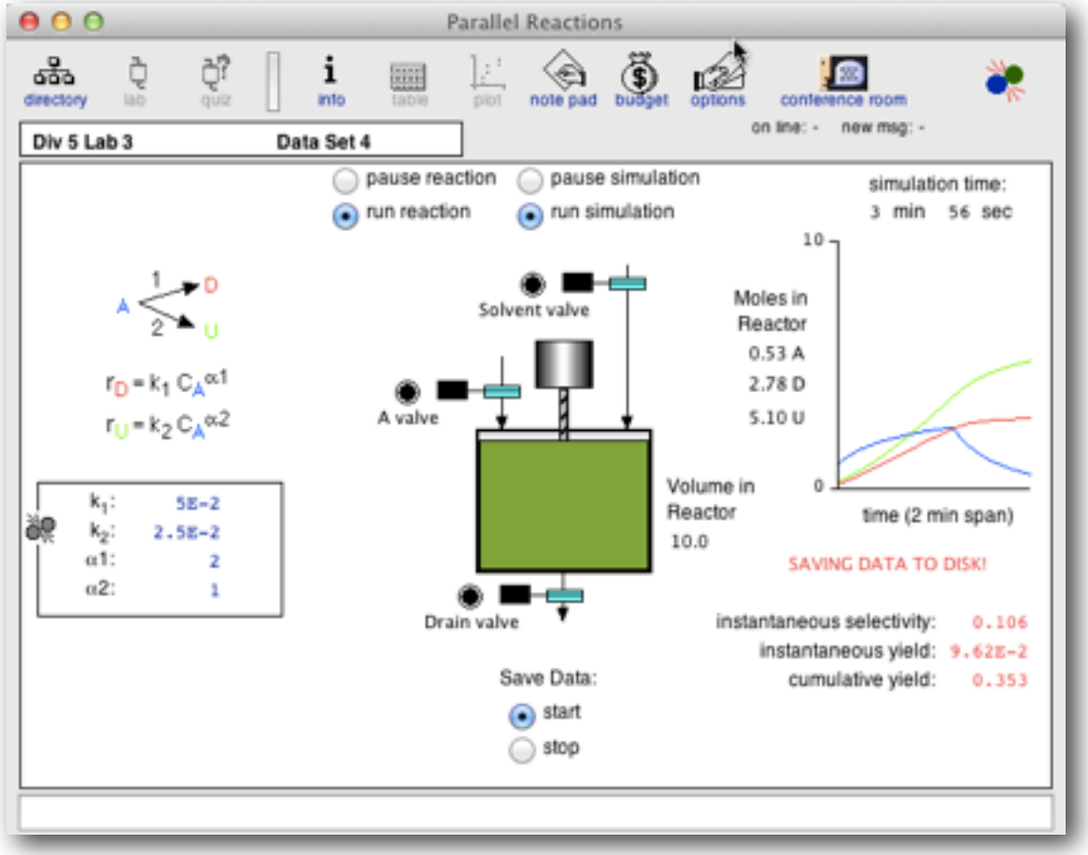
SimzLab Cours en génie chimique :

- Reactor Lab : simulations de réacteurs chimiques
- Pure Water Lab : purification de l'eau et la réutilisation
- Control Lab : simulations de contrôle de processus

Caractéristiques :

- Des simulations interactives pour engager les élèves dans un apprentissage actif
- Utilisé par des étudiants de plus de 100 pays depuis 2003
- La puissance d'une application de bureau
- Accès Internet pour utiliser en ligne ou hors ligne sur un PC Windows ou Mac

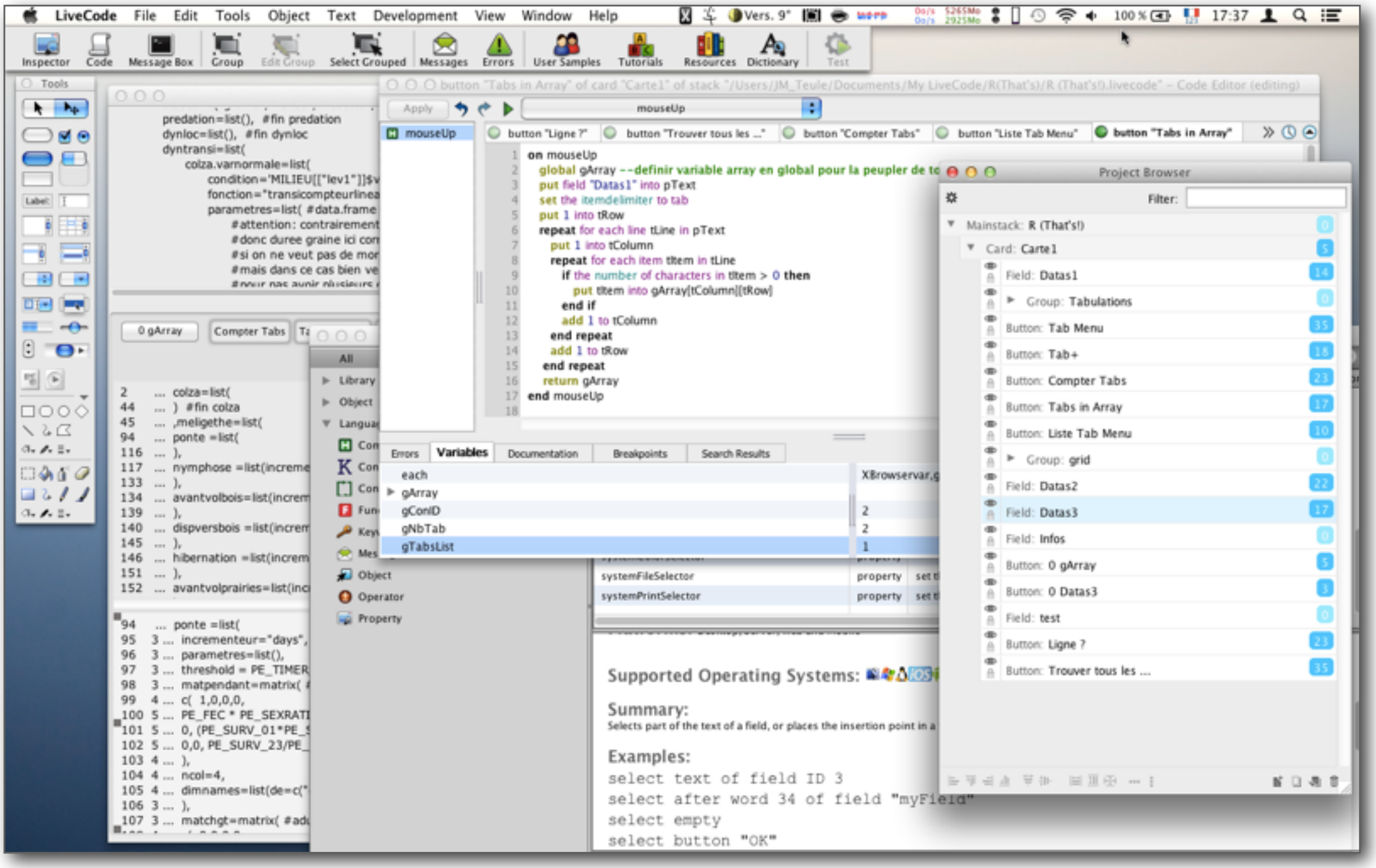




LiveCode, quelques exemples d'applications

Last but not least, LiveCode himself !

Eh oui ! L'IDE de **LiveCode** est totalement réalisé et fonctionne en **LiveCode**. Par défaut, ses composants sont masqués mais peuvent être révélés et modifiés (très fortement déconseillé aux débutants !).



Pour conclure

Tout ceci n'est qu'un aperçu de **LiveCode**. Plein de choses n'ont pas été dites et ne peuvent l'être ici.

Si vous cherchez à :

- développer des utilitaires, des interfaces pour vos applications personnelles,
- développer des prototypes ou plus encore pour des projets informatiques plus importants,
- développer des interfaces de traitement de vos données créer une interface pour des applications qui en sont dépourvues,
- créer des maquettes fonctionnelles pour une recherche de financement,
- créer des cours réellement interactifs,
- créer des interfaces de terrain sur des mobiles,

alors **LiveCode** est peut être la solution.

Un dernier point : ce document commence par le témoignage de Robert Cailliau plus qu'enthousiaste.

J'ai voulu qu'il se conclue par celui de Marie Gosme, qui travaillait au sein l'unité **Agronomie** et commençait alors à peine à expérimenter **LiveCode**.

Pour

LiveCode permet de faire des interfaces graphiques qui ont l'air pro, en adaptant automatiquement l'aspect au système d'exploitation de l'utilisateur (forme et couleur des boutons, interface de navigation de fichier etc..).

Il propose tous les éléments imaginables dans une interface : boutons, champs de saisie, listes déroulantes, menus pop-ups, tableau dont on peut sélectionner les lignes et trier les colonnes etc...).

Il suffit d'organiser ces boutons comme on veut dans une fenêtre puis leur attribuer une action sous forme de script qui sera déclenché lorsque l'utilisateur cliquera dessus. On peut créer ainsi plusieurs "pages" d'interfaces et éventuellement plusieurs fenêtres ouvertes simultanément.

L'utilisation de l'interface du logiciel est simple et permet de passer "en live" (d'où le nom de livecode) du mode "création" au mode "utilisation" (i.e. quand on clique sur un bouton en mode "création", on peut le déplacer, renommer, modifier le script correspondant, quand on clique dessus en mode "utilisation", ça déclenche le comportement associé au bouton).

Les tutoriels sont bien faits, en particuliers ceux qui sont eux-mêmes faits avec **LiveCode** : il suffit de passer du mode utili-

sation au mode création pour jeter un coup d'œil sur les scripts qui correspondent aux exemples présentés (et en réutiliser des bouts pour sa propre application).

Contre

Le langage de programmation utilisé pour écrire les scripts des différents boutons n'est (à mon avis) pas du tout optimal pour la programmation : en voulant avoir un langage qui ressemble au langage humain (en l'occurrence de l'anglais), on obtient un truc intermédiaire, qui n'est ni naturel ni rigoureux. Il y a beaucoup de synonymes (deux mots différents qui veulent dire exactement la même chose, par exemple "***cr***" et "***return***") mais aussi beaucoup de faux amis (des mots qui sont synonymes dans le langage courant mais qui correspondent à des fonctions différentes dans **LiveCode**, par exemple "***clone***" et "***copy***").

Les "pros" conseillent souvent sur le forum d'essayer d'écrire ce qu'on veut en anglais : souvent ça marche, mais pas toujours (et alors on ne sait pas pourquoi). D'ailleurs attention : toutes les commandes ne sont pas expliquées dans l'aide : soit on a de la chance et on a écrit par hasard ce qu'il fallait, soit il faut tâtonner jusqu'à trouver la bonne formulation à partir des exemples trouvés sur les forums. L'utilisation des forums peut parfois être difficile lorsqu'on cherche des infos sur des mots du langage qui sont aussi des mots très courants en anglais (par exemple "***it***").

Enfin il faut bien se dire que ***LiveCode*** n'est pas fait pour manipuler des données : les "***arrays***" ne permettent pas d'accéder facilement aux éléments "transversalement" (par exemple aux colonnes d'une matrice); il n'y a pas de "vectorisation" (au sens de R) des calculs donc il faut faire une boucle explicitement même pour ajouter 1 à tous les éléments d'un ensemble de chiffres.

Conclusion : j'utilise **LiveCode** car il fait des choses que je ne pourrais pas faire autrement, mais j'aurais préféré que les créateurs du logiciel utilisent R comme langage de script!

***Note JMT** : un commentaire à ce commentaire : la comparaison faite, ci-dessus, avec le langage **R** a un biais selon moi. La portée des deux langages n'est pas la même. Là ou **LiveCode** est généraliste avec comme objet principal la réalisation d'une interface, **R** est spécialisé dans le calcul statistique et n'a pas à s'occuper d'interface. Il est donc normal que leurs capacités ne soient pas identiques et que **R** soit bien meilleur pour ce qu'il a à faire.*

*Ce qui n'empêche pas de pouvoir utiliser **R** depuis et avec **LiveCode** pour créer une interface utilisateur sur la base de **R** par exemple !*