Visual Basic Entre MS-DOS et Windows

Pour IBM et compatibles

Tous les regards sont tournés vers Windows, et toc ! Microsoft agit là où on ne l'attendait pas : il édite Visual Basic pour MS-DOS. Cette version du nouveau langage de programmation fait coup double. Elle s'offre le luxe d'être compatible avec Visual Basic pour Windows – en s'appuyant , comme lui, sur la programmation événementielle –, et aussi avec les logiciels plus classiques pour MS-DOS, Quick Basic et PDS 7. Pour les amateurs comme pour les spécialistes de MS-DOS, voici la bonne occasion de passer en douceur à Windows.

Par Patrick Zémour

Microsoft a bien joué ! Son P.-D.G. et fondateur, Bill Gates, y est sans doute pour beaucoup. Ardent défenseur du Basic, l'homme veut voir son langage fétiche continuer de vivre pour suivre un destin hors du commun : devenir le langage universel de Windows. Certes, nous n'en sommes pas encore là, mais en annonçant la sortie de Visual Basic pour MS-DOS, l'éditeur américain offre un remarquable outil pour le développement rapide d'applications MS-DOS à l'aspect professionnel... et donne aux programmeurs une raison de plus de rester fidèles au Basic, même lorsqu'ils décideront de passer à Windows.

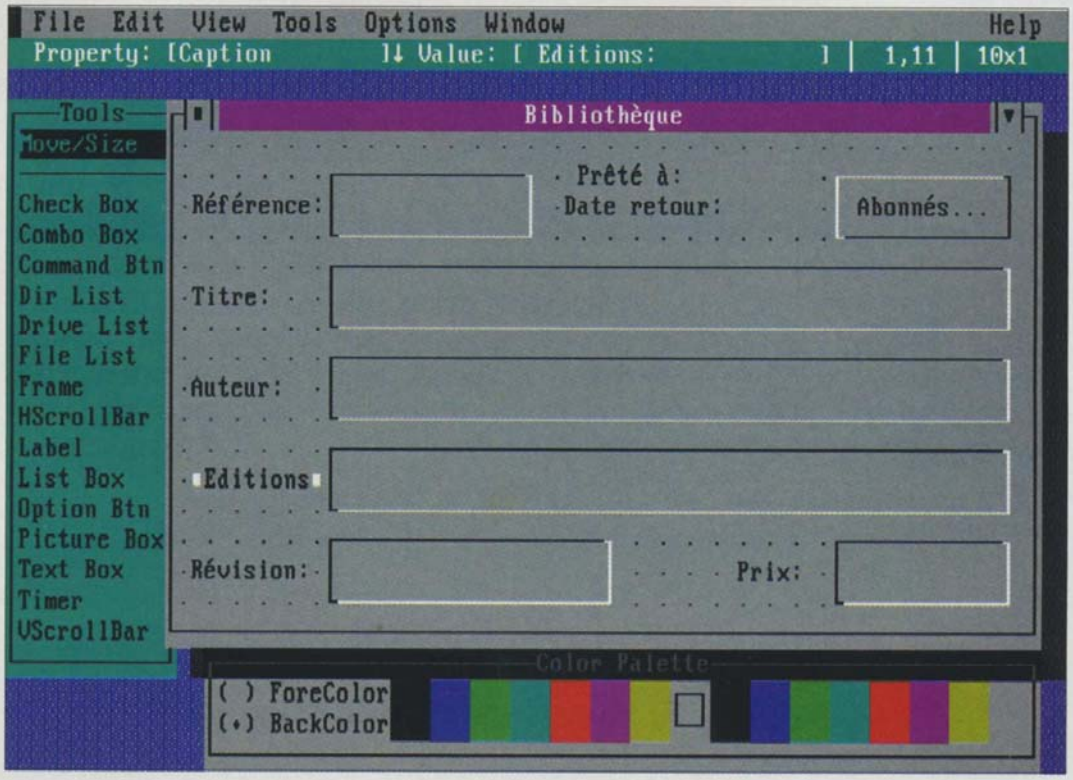
D'un côté, il est parfaitement compatible avec Visual Basic pour Windows (voir *SVM* n° 84), car il en reprend les principes de la programmation événementielle. De l'autre, il reste compatible avec ses aînés pour MS-DOS, Quick Basic et le très professionnel PDS 7, car il garde la possibilité de programmer selon les méthodes en usage auparavant. Dans cet environnement hybride pour MS-DOS, les anciennes applications écrites en Quick Basic pourront donc encore évoluer, tandis que les nouvelles gagneront à être développées selon les principes de la programmation événementielle. La stratégie est habile.

Visual Basic pour MS-DOS, testé ici dans une pré-version américaine, sortira dans le courant du mois de septembre en français. Il sera proposé en deux versions, l'une Standard, l'autre plus étoffée, baptisée édition Professionnelle.

La nouvelle recette pour les applications

Avec cette version de Visual Basic, Microsoft offre l'occasion aux amateurs des précédents Basic travaillant sous MS-DOS de découvrir sa nouvelle recette de fabrication des applications.

LOGICIEL TESTÉ
Pré-version de Visual Basic pour MS-DOS, en anglais. Édité par Microsoft. Version définitive et en français prévue pour début septembre. Ordre de prix : de 1 500 F à 2 000 F HT pour l'édition Standard, et de 3 000 à 4 000 F HT pour l'édition Professionnelle.



Visual Basic pour MS-DOS présente une interface utilisateur très proche de la version pour Windows.

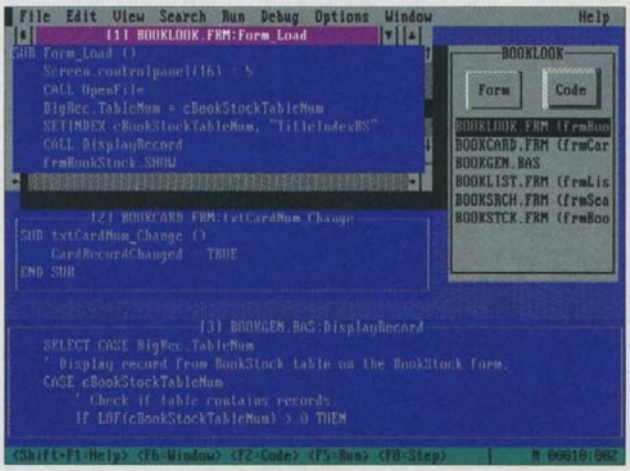
Plutôt que de décrire le contenu de l'écran à l'intérieur même du programme, le développeur "dessine" ici l'interface utilisateur grâce au Créateur de feuilles – une feuille étant, dans Visual Basic, une fenêtre ou une boîte de dialogue. Pour travailler rapidement, la souris est fortement recommandée : on pioche dans une palette d'outils les différents objets, encore appelés contrôles, qui s'afficheront dans la feuille. Parmi la quinzaine que l'on peut placer ainsi, citons les boutons, cases à cocher, cases d'options, listes, listes modifiables (pour sélectionner un élément dans une liste déroulante ou taper directement son choix), cadres, barres de défilement et zones de saisie. S'y ajoute un objet Minuterie, capable, lui, d'exécuter une action à un intervalle donné. Autres contrôles très précieux, les listes de type disque, répertoire et fichier, avec lesquels il est possible de créer, par exemple, une boîte de dialogue d'ouverture ou d'enregistrement de fichier, pratiquement sans programmer.

Signalons enfin que, contrairement à ce que laisse supposer son nom, tiré de la version Windows, l'objet appelé Zone d'image ne peut contenir des graphiques, mais plus modestement des textes courant sur plusieurs lignes.

Dans le Créateur de feuilles, les objets peuvent être déplacés et redimensionnés à loisir, et il est possible, sans même avoir commencé à programmer, de modifier leur apparence et leur comportement. Il suffit, après avoir cliqué sur un contrôle pour le sélectionner, de le déplacer ou modifier ses caractéristiques grâce à la barre de propriétés située sous le menu. Car chaque objet possède des caractéristiques qui lui sont propres, tels que son nom bien sûr, mais aussi sa couleur, sa visibilité ou non à l'écran, l'affichage d'un libellé particulier, etc., jusqu'à l'apparence du pointeur de la souris lorsqu'il glissera sur cet objet.

Toutes ces propriétés pourront bien sûr être modifiées durant l'exécution du programme, mais il va sans dire qu'avec cette méthode interactive de définition de l'affichage, le temps de développement se trouve déjà considérablement réduit. D'autant que la même fenêtre peut être utilisée dans plusieurs programmes distincts, et que diverses boîtes de dialogues courantes sont fournies prêtes à l'emploi (ouverture et enregistrement de fichiers, options d'impression, recherche/remplacement, etc.).

D'autre part, si l'on duplique un objet par copier/coller, le logiciel propose de l'indexer. Dans le programme, on fera référence à ces objets en faisant suivre leur nom par un indice (objet(1), objet(2)...) comme s'il s'agissait d'un tableau.



L'environnement du développement : pour programmer "classique" ou par événements.

Par contre, et c'est dommage, l'indexation des feuilles elles-mêmes n'a pas été prévue. Le Créateur des feuilles comporte également un éditeur de menus avec lequel on définit un menu déroulant associé à la fenêtre.

Ajoutons que de nouveaux contrôles, achetés ou créés par l'utilisateur de la version Professionnelle (*voir encadré*), peuvent étoffer la quinzaine d'objets déjà disponibles.

Un objet répond à des événements

Une fois le contenu des fenêtres créé, on passe à l'environnement de programmation dans lequel on bâtit véritablement son projet. L'environnement permet de saisir les instructions Basic, d'exécuter le programme et de le déboguer. C'est là que l'on définit le comportement de l'application, en précisant par exemple quelle action le programme doit entreprendre quand l'utilisateur cliquera sur un bouton ou lorsqu'il choisira un nouvel élément dans une liste.

Chaque objet peut ainsi répondre à un jeu prédéfini d'événements (clic de souris, double-clic, appui sur une touche, changement du contenu d'une zone de saisie...), qui diffèrera selon la nature de l'objet. Selon les impératifs de l'application, le programmeur liera à certains de ces événements une séquence d'instructions Basic.

Un projet-type se compose d'une ou plusieurs feuilles (et de leurs contrôles) et de modules Basic, voire de librairies complémentaires. En appuyant sur la touche F12, Visual Basic montre, pour chaque feuille, la liste des objets qu'elle contient ainsi que le nom de chaque procédure événementielle rattachée aux objets. C'est au programmeur de décider quelle procédure il doit "remplir" pour que son application fonctionne.

Prenons un exemple : comme la plupart des contrôles, l'objet Bouton détecte l'événement Click (de souris) et lui réserve une procédure. Par défaut, cette procédure est vide et un clic sur le bouton ne produit donc aucun effet ; l'action n'aura d'effet que si le programmeur place des instructions Basic dans la procédure Bouton.Click.

Bref... le développeur abandonne la programmation classique, où toutes les instructions s'enchaînent les unes à la suite des autres, au profit d'une programmation événementielle ; les ordres Basic sont regroupés dans des procédures qui ne s'exécutent qu'en réponse à un événement produit par l'utilisateur ou par l'ordinateur lui-même.

En d'autres termes, le programmeur n'a plus à vérifier si l'utilisateur va presser le bouton OK, il se contente de prévoir quelle action mener quand Visual Basic lui indique que cet événement vient de se produire.

Cela étant, toutes les instructions composant un programme ne sont pas obligatoirement rattachées à l'un des

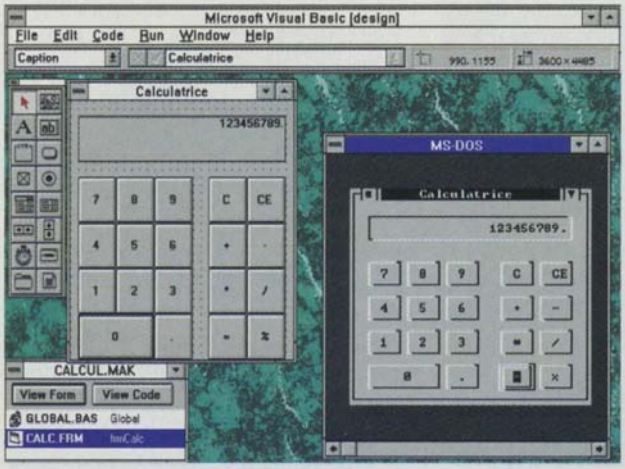
AIDE

La chasse aux bogues est ouverte

Option Explicit, une instruction noyée dans le flot des nouveautés – par rapport au Basic traditionnel – de Visual Basic pour MS-DOS, passe presque inaperçue. Pourtant, elle se révélera particulièrement utile. En effet, cette simple petite instruction épargnera aux programmeurs de longues heures passées à traquer les vilaines bogues qui se glissent dans les applications.

On le sait, avec le Basic comme avec la plupart des langages, on stocke les données dans des variables. Mais le Basic ayant été conçu pour les débutants, il est beaucoup plus souple que ses confrères et n'oblige pas à déclarer les variables avant de les utiliser. Revers de la médaille, si l'on crée une variable "Famille", puis que l'on y fait référence sous le nom de "Faille" – une faute de frappe est si vite arrivée –, le Basic ne détecte pas d'erreur. Il crée simplement une nouvelle variable appelée "Faille", et le programme ne fonctionne évidemment pas correctement... Malgré ce risque fréquent, dans la communauté des développeurs, nombreux sont les informaticiens à défendre la déclaration facultative des variables, car elle leur évite de taper beaucoup de lignes jugées inutiles. Avec Option Explicit, ce genre de mésaventure devient impossible, car à l'instar du langage C ou du Pascal, Visual Basic s'arrête dès qu'il rencontre une variable non déclarée.

Fervents défenseurs du Basic traditionnel, inutile de manifester colère et indignation... comme son nom l'indique, la nouvelle instruction est optionnelle !



Un traducteur convertit les programmes d'un environnement à l'autre.

événements prédéfinis d'un objet, mais peuvent être réunies dans des procédures générales appelées depuis n'importe quel objet.

Pour le débutant, cette méthode de programmation est un peu déroutante, et il aura sans doute souvent recours au débogueur pour tester ses programmes. Heureusement, cet indispensable outil est bien plus puissant que celui présent dans la version Windows. Un mode Historique conserve ainsi les vingt dernières instructions exécutées, que l'on peut ensuite parcourir. Plus conventionnel, en plaçant des points d'arrêts sur les instructions Basic, on a toute latitude pour vérifier le contenu des variables ou suivre le déroulement du programme pas à pas. Visual Basic peut également interrompre l'exécution dès qu'une expression précisée par l'utilisateur devient vraie.

Exclusivement en mode caractère

Les amateurs de Visual Basic pour Windows l'auront compris, la version MS-DOS respecte les principes qui ont fait le succès du langage pour Windows. Avec une différence notable cependant : Visual Basic pour MS-DOS, autant que les applications à base de fenêtres conçues avec lui, est ici en mode caractère. Pour être exact, rien n'empêche un programme de basculer du mode texte au mode graphique, mais les fenêtres ne s'affichent que dans le premier mode.

La ressemblance avec Windows n'en est pas moins fidèle, puisque les fenêtres disposent aussi de boutons de réduction et d'agrandissement, et d'un menu Système que l'on appelle en cliquant sur une pastille située en haut à gauche de la fenêtre. Si une fenêtre minimisée n'apparaît pas sous forme d'icône comme dans l'environnement graphique, elle se réduit néanmoins à un petit cadre rangé en bas de l'écran. De même, en jouant sur la couleur des caractères semi-graphiques des IBM et compatibles, Microsoft a pu donner un semblant de relief aux objets affichés.

Afin que son langage puisse s'exécuter sur un modeste ordinateur équipé de 640 Ko de mémoire vive, l'éditeur a scindé son logiciel en deux programmes distincts, le fameux Créateur de feuilles d'une part, et l'environnement de programmation et de débogage, d'autre part. L'intention est louable, mais cette division du logiciel, qui n'existe pas dans Visual Basic pour Windows, s'avère à la longue franchement contraignante, car elle oblige à de fréquents allers-retours d'une application à l'autre.

Le point déterminant pour le succès de Visual Basic, c'est la compatibilité. Rappelons que l'objectif de Microsoft était double. Tout en étant compatible avec son équivalent pour Windows, Visual Basic ne devait pas pour autant couper les ponts avec le Quick Basic et le PDS 7 (Basic professionnel de Microsoft), qu'il est censé remplacer. Résultat de ce compromis : contrairement à Visual pour Windows, l'usage du Créateur de feuilles et de la programmation par événements n'est pas une obligation pour écrire une application.

D'un environnement à l'autre

Ainsi, un programme écrit avec le QBasic (livré avec MS-DOS 5.0), Quick Basic ou avec le PDS 7 reste entièrement compréhensible par Visual pour MS-DOS (il faut cependant posséder les fichiers "source" des librairies appelées par le programme). Et l'on peut continuer à développer des applications selon les principes de la programmation classique, voire panacher les deux méthodes. Mais attention, il ne faut pas espérer exécuter ces applications dans Visual Windows sans un long et profond lifting.

En revanche, si l'on a recours uniquement au Créateur de feuilles pour dessiner l'interface utilisateur de l'application, on adopte du même coup la programmation événementielle, et on s'assure alors d'une excellente compatibilité avec le langage pour Windows L'utilisateur, de même que les programmes, passera donc d'un environnement à l'autre sans problème – la conversion des lettres accentuées n'étant toutefois pas encore correctement assumée par la pré-version testée.

Un traducteur est fourni sous la forme d'une application Windows, qui se charge de convertir les programmes d'un environnement à l'autre, en traduisant notamment les coordonnées des objets affichés à l'écran du mode caractère au mode graphique et inversement. Bien sûr, de Windows vers MS-DOS, toute référence à un objet graphique, image en mode point ou icône est ignorée durant la traduction. Et quelques retouches seront parfois nécessaires. Mais le gros-œuvre est fait !

Grâce à sa double compatibilité et aux attraits de la programmation événementielle, Visual Basic pour MS-DOS devrait rapidement s'imposer comme l'outil de référence pour les programmeurs qui affectionnent ce langage. Mais nous connaissons, nous, un moyen infaillible pour accélérer la percée de Visual Basic... le livrer, en lieu et place du QBasic, avec la prochaine version de MS-DOS ! ●

EXPERT

Professionnels ne pas s'abstenir

Microsoft proposera deux versions de son Visual Basic pour MS-DOS : une version Standard, destinée aux amateurs, et une édition Professionnelle qui inclut plusieurs outils et bibliothèques présents dans le Microsoft Basic PDS 7, auxquels s'ajoutent quelques nouveautés. Côté performances, le compilateur de la version Pro pourra par exemple créer des programmes optimisés pour les processeurs 286 ou 386, tandis que sur un ordinateur non équipé d'un coprocesseur mathématique, une librairie spéciale exécutera les calculs bien plus rapidement que l'édition Standard. Par ailleurs, lorsqu'on crée une grosse application qui ne tient pas dans les 640 Ko de mémoire conventionnelle, il est indispensable de scinder le programme en plusieurs modules ; classiquement, par un mécanisme automatique, le programme charge et décharge les modules entre disque et mémoire vive. Selon la configuration de l'ordinateur, le gestionnaire Move fourni par Microsoft saura loger ces modules en mémoire étendue ou paginée plutôt que sur disque, d'où un gain de vitesse appréciable. Autres fonctions réservées aux professionnels, la gestion de fichiers séquentiels-indexés, ainsi que plusieurs boîtes à outils. On trouve notamment un ensemble de fonctions financières et mathématiques, un kit graphique pour tracer des camemberts, histogrammes et autres diagrammes linéaires, et un kit facilitant la réalisation d'aides hypertextes. Des utilitaires permettront enfin au programmeur chevronné de créer ses propres objets, qu'il pourra ensuite placer dans les fenêtres et boîtes de dialogues des applications, aux côtés des objets fournis par Microsoft. La distinction entre éditions Standard et Professionnelle se retrouvera également dans Windows avant la fin de l'année, quand sortira la version 2.0 du Visual Basic. La version professionnelle pour Windows comprendra alors la plupart des fonctions incluses dans celle pour MS-DOS.