***Chapitre 1***

# Introduction à la programmation Visual Basic

## *1.1 Introduction*

*La construction d'une application Visuel Basic doit être réalisée conformément aux étapes suivantes :*

- *Création de l'interface à l'aide d'un générateur interactif d'interfaces. Il s'agit de placer les interacteurs ou contrôles de l'interface dans les fenêtres de l'application. Cette opération est réalisée par manipulation directe à partir d'une boite à outils.*
- *Tester le comportement de l'interface.*
- *Ensuite, définir le comportement de l'application en associant les procédures adéquates aux différents événements pouvant être générée par les interacteurs de l'interface. Ce mode de développement rentre dans le cadre de la programmation événementielle.*

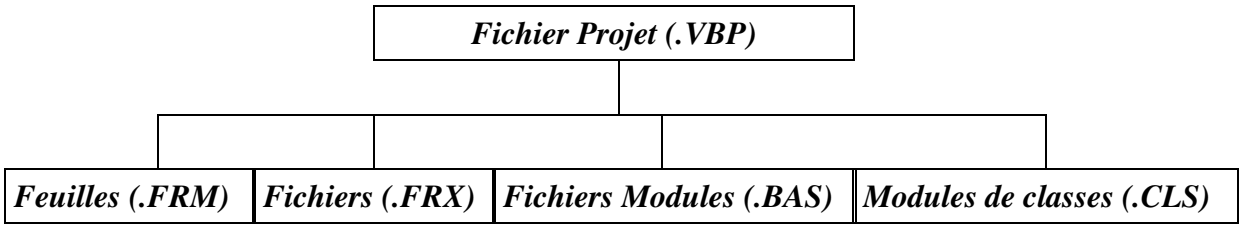
## *1.2 Projet VB*

*Un projet VB se compose d'un certain ensemble de fichiers qui constituent toute l'application. Ces constituants sont les suivants :*

- *Le projet lui même qui consiste en un fichier texte d'extension* ***.vbp (visual Basic Project)*** *contenant des informations sur le type du projet, les constituants de ce projet ainsi que des références aux différentes bibliothèques qui devront être compilées avec l'application. On peut citer des fichiers* ***.OCX, .DLL*** *ou autres références comme la librairie des Objets d'accès aux bases de données (exemple : Microsoft DAO 3.5 Object Library).*

  *Par défaut, lorsqu'on crée une nouvelle application, un projet est alors créée automatiquement :* ***Projet1.vbp***
- *Des fichiers textes d'extension* ***.frm (FORM)*** *qui constituent les feuilles (ou les fenêtres) de l'application. Un projet contient au moins une feuille (par défaut Form1.frm) et il peut contenir plusieurs feuilles. Les constituants minimaux d'une application VB sont un fichier projet et un fichier feuille* ***(Projet1.vbp + Form1.frm)****.*
- *Des fichiers binaires d'extension .frx. Chaque fichier .frx (exemple Form1.frx) est associé à un fichier .frm qui porte le même nom (exemple Form1.frm). les fichiers .frx contiennent les données binaires des feuilles comme les images, les icônes etc… Ce qui veut dire que si la feuille ne contient pas de données binaires il n'y aurait pas de fichier .frx associé.*
- *Des fichiers modules qui sont des fichiers texte d'extension* ***.bas (Basic).*** *Ces fichiers sont très utiles dans la pratique puisque c'est le seul endroit où l'on peut mettre les données, structures et fonctions communes de l'application, accessibles par tous les constituants du projet. Un fichier module n'a pas de représentation visuelle, il ne lui est associé aucune fenêtre de l'interface.*
- *Des modules de classes qui sont des fichiers textes d'extension .cls. Ces modules, comme les fichiers modules, ne sont pas associés à des feuilles. Il ressemble aux modules .bas avec la différence que ceux-ci permettent la création de classes à partir desquelles on peut instancier des objets, et rentre alors dans le cadre de la programmation orientée objets.*

*Un projet VB se présente alors comme suit :*




*L'exemple suivant présente un aperçu d'un fichier Projet .VBP. Nous pouvons noter les lignes qui commencent avec le mot clé Form ou le mot clé Module, permettant de préciser les feuilles et les modules qui constituent un projet.*

*Type=Exe*

*Form=Form1.frm*

*Module=Module1; Module1.bas*

*Reference=*\G{00025E01-0000-0000-C000-000000000046}#4.0#0#..\..\ProgramFiles\Fichiers communs\Microsoft SharE:\Program Files\#Microsoft DAO 3.5 Object Library*

*Form= Form2.frm*

*Form= Form3.frm*

*Object= {F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.1#0; COMDLG32.OCX*

*Form= Form4.frm*

*IconForm="Form1"*

*Startup="Form1"*

*HelpFile=""*

*Command32=""*

*Name="Projet1"*

*…*

***Remarque :***

*Les Fichier .DLL (Dynamic Link Librairies : librairies à liaison dynamique) et leurs versions évoluées les fichiers .OCX (les ActiveX) sont des bibliothèques qui contiennent un certains ensemble de composants pouvant être intégrés dans les applications utilisateur. Leur présence est alors obligatoire aussi bien lors de la réalisation (on dit la conception) de l'application que lors de son exécution. Ces bibliothèques sont alors, elles aussi, référencées dans le projet VB.*

*Nous signalons que certaines de ces bibliothèques font partie du système, alors que d'autres sont copiées sur le disque lors de l'installation de Visuel Basic. Généralement ces bibliothèques se trouvent dans le sou répertoire SYSTEM du répertoire d'installation de Windows.*

## *1.3 Notion de contrôle*

*La feuille (ou Form) est le composant de base d'une application VB. Une feuille est alors composée d'un ensemble d'objets graphiques qui permettent l'interaction avec l'utilisateur de l'application. Ce dernier clique sur des boutons, coche des cases, sélectionne des options de menu, écrit sur des zones de texte etc… Ces éléments qui composent la feuille, ainsi que la feuille elle-même, constituent des* ***objets graphiques*** *qu'on appelle interacteurs ou* ***contrôles.***

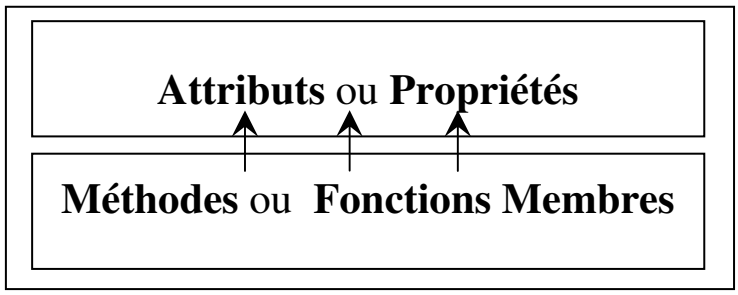
### *1.3.1 Qu'est ce qu'un objet*

*Une structuration classique d'un programme consiste en une structuration à deux niveaux : les données d'une part et le code d'une autre part. Ainsi les données qui décrivent ou caractérisent une même entité sont regroupées ensemble dans une même structure de donnée : un enregistrement ou un tableau. De la même manière des instructions réalisant ensemble une tache bien définie et complète sont regroupées dans une même procédure ou fonction.*

*La Programmation Orientée Objets (**POO**) consiste en une structuration de haut niveau. Il s'agit de regrouper ensemble les données et toutes le procédures qui permettent la gestion de ces données. On obtient alors des entités comportant à la fois un ensemble de données et une liste de procédures et de fonctions pour manipuler ces données. La structure ainsi obtenue est appelée : **Objet**.*

*Un objet est alors une généralisation de la notion d'enregistrement. Il est composé de deux parties :*

- *Une partie statique (fixe) composée de la liste des données de l'objet. On les appelle : **Attributs ou Propriétés**.*
- *Une partie dynamique qui décrit le comportement ou les fonctionnalités de l'objet. Elle est constituée de l'ensemble des procédures et des fonctions qui permettent à l'utilisateur de configurer et de manipuler l'objet. Ainsi les données ne sont généralement pas accessibles directement mais à travers les procédures et les fonctions de l'objet. Celles-ci sont appelées : **Méthodes ou Fonctions Membres**.*




### *1.3.2 Définition d'un contrôle*

*Un Contrôle est un Objet d'interface. Il est utilisé généralement sur l'interface d'une application graphique pour fournir un certain ensemble de fonctionnalités. Un contrôle est alors constitué d'un ensemble de **Propriétés** et de **Méthodes** (puisque c'est un Objet). En plus, il dispose d'un ensemble de procédures particulières qu'on appelle des **Evénements**. Ces 3 entités d'un contrôle sont définies comme suit :*

***1- Les Propriétés :***

*Chaque contrôle dispose d'un ensemble de propriétés qui le caractérise. Parmi les propriétés communes aux contrôles, on peut citer les suivantes :*

- ***Name** : le Nom du contrôle*
- ***Top, Left, Width, Height** : qui sont les propriétés de position et de dimension dont dispose tous les contrôles graphiques.*
- ***Font** : qui permet de préciser la police et autres caractéristiques du texte d'un contrôle (taille, gras, soulignement,…).*
- ***BackColor** : couleur de fond du contrôle.*

### *2- Les Méthodes :*

*Ce sont des procédures qui permettent de configurer ou d'appliquer un traitement particulier à un contrôle. En effet, certaines actions ne peuvent être obtenues à l'aide des propriétés mais uniquement par l'intermédiaire des méthodes.*

*Les méthodes d'un objet sont accessibles de la même manière que les propriétés par l'intermédiaire du sélecteur point «. » :*

*Contrôle.Méthode*

***Exemples de méthodes :***

- ***SetFocus** : elle permet de déplacer le curseur ou de donner la main (donner le focus) à un contrôle :*

  *C1.SetFocus ➔ donne le focus au contrôle C1.*
- ***Clear** : permet d'effacer le contenu d'un Contrôle (exemple un contrôle ListBox)*
- ***Show** : permet de visualiser une feuille :*

  *Form2.Show*

### *3- Les Evénements :*

*Un événement est une procédure qui n'est pas appelée explicitement par le programme (ou le programmeur), mais qui se déclenche automatiquement en réponse à une action externe lors de l'exécution de l'application. Cette action peut être par exemple, un click sur un bouton de la souris, sélection d'une option de menu ou d'une liste, expiration d'un temporisateur, etc…*

*Exemples :*

- ***Click** : se déclenche lorsque l'utilisateur clique sur un contrôle de l'interface.*
- ***KeyPress** : se déclenche lorsque l'utilisateur tape sur une touche du clavier.*
- ***Load** : se déclenche lors du chargement d'une feuille.*

***Remarques :***

1- *Les contrôles Visual Basic disposent en général de plusieurs événements qui surveillent en détail toutes les actions utilisateur dur l'interface.*

2- *Il existe certains événements qui ne sont pas reliés à des actions utilisateurs, mais des actions du système ou d'autres applications. On peut citer le cas du temporisateur (le contrôle Timer).*

3- *Toutes les procédures de traitement des événements sont par défaut vide (aucun code). Pour réaliser une action quelconque en réponse à un événement, il faut écrire le code correspondant à l'intérieur de la procédure de traitement de l'événement. Il s'agit de la programmation événementielle qui se base sur la programmation des événements associés aux différents contrôles de l'interface.*

## *1.4 Programmation Evénementielle*

*Après avoir configuré l'interface, le programmeur doit associer le comportement adéquat au contrôle adéquat. Cela consiste à répondre à la question :*

***Quel événement** de **quel contrôle** faut-il programmer ?*

*Ainsi, à chaque événement à intercepter, il faut développer la procédure de traitement d'événement associée. En effet, pour un contrôle nommé **Control**, la procédure de traitement de son événement nommé **Event** aura le nom **Control_Event** et la structure suivante :*

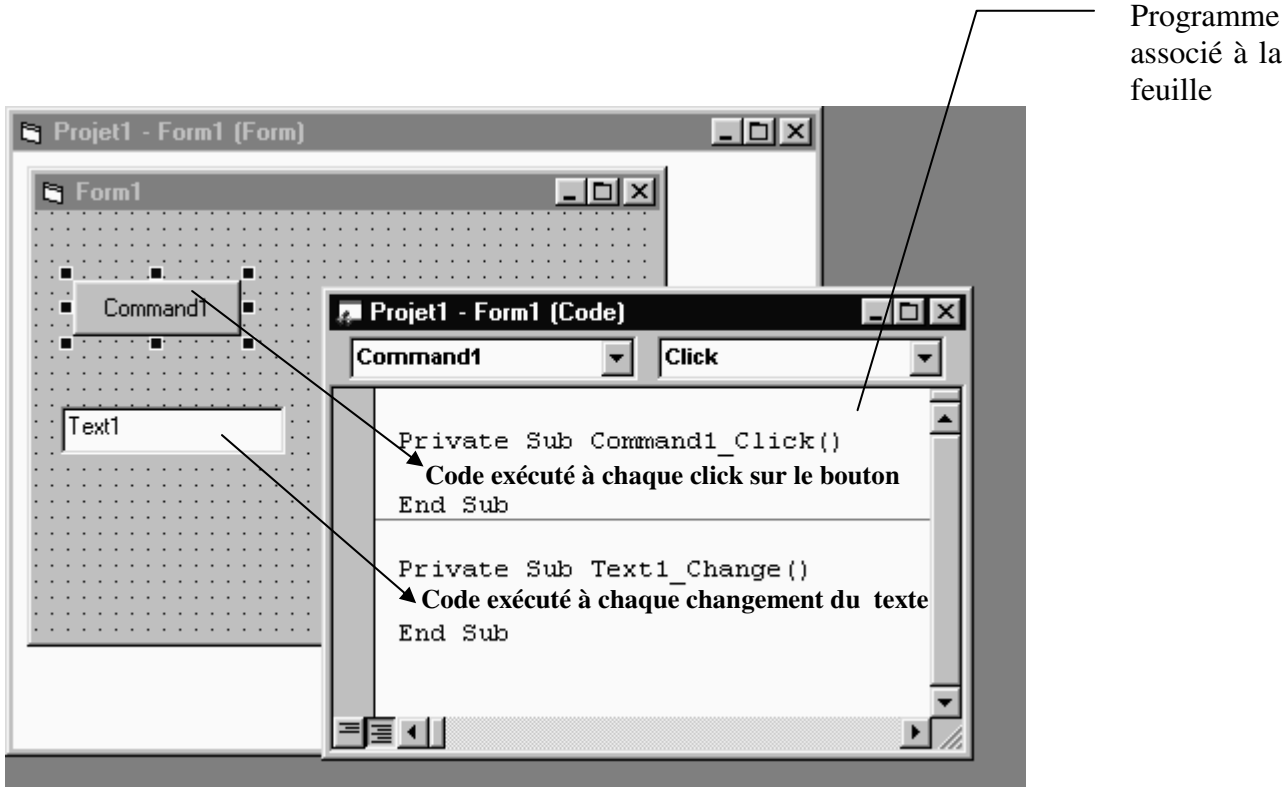
```
Private Sub Control_Event (paramètres éventuels)

End Sub
```

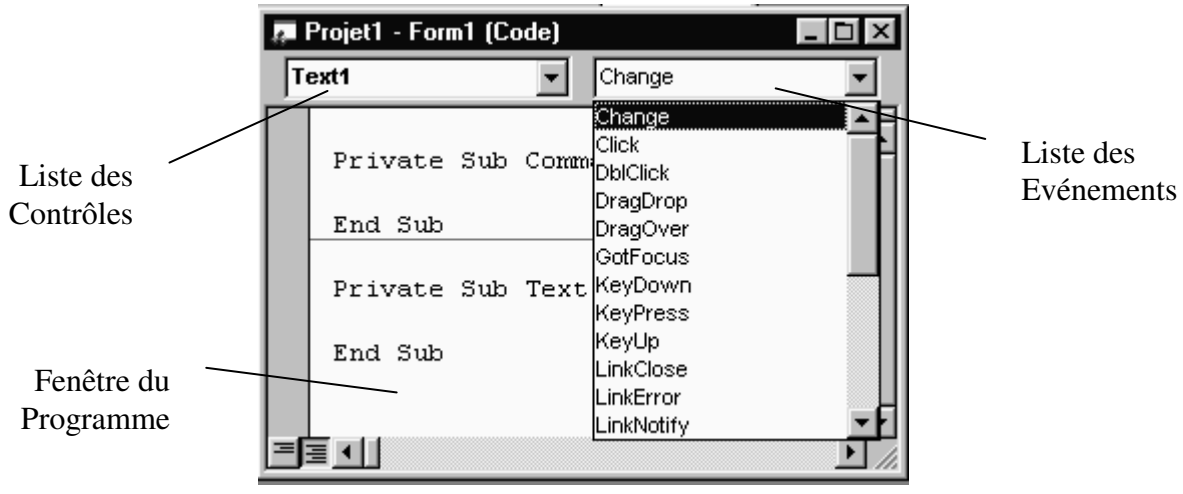
*On remarque que la syntaxe d'ecriture des procédures et fonctions est dévelopée dans le chapitre suivant.*

***Remarques :***

*1- Un double-click sur le contrôle permet de créer automatiquement la procédure de traitement d'un événement par défaut du contrôle qu'il suffit de remplir avec le comportement désiré.*




*2- A chaque contrôle est associé une liste d'événements qu'il est possible d'intercepter. Si l'événement choisi par défaut ne convient pas, il est donc possible de sélectionner parmi la liste l'événement adéquat (voir figure suivante) :*



## *1.5 Environnement de Développement Visual Basic*