De Visual Basic à Gambas

Première partie

Préambule :

Beaucoup de gens sont abonnés à Microsoft pour l'utilisation de leur ordinateur. Déjà par le système d'exploitation, et son célèbre « Windows », et souvent, par l'utilisation de nombreux logiciels (payants), de la même marque. Certains ont aussi le souhait de développer leurs propres programmes – que ce soit pour des raisons particulières ou simplement ludiques, et utilisent (ou utilisaient) le langage Visual Basic et son éditeur spécialisé (payant également).
Beaucoup de gens aussi se sont lassés des coûts pharamineux et du mercantilisme commercial de la « marque vedette », et ont radicalement évolué vers le Logiciel Libre. Ainsi, la migration sous Linux convient aux personnes libres, et qui sont prêtes à faire un minimum d'effort d'adaptation, car tout changement implique une remise en cause de ses habitudes.
Quand j'étais sous Microsoft, j'ai trouvé un jour un livre formidable qui s'intitule « Visual Basic 6 – la programmation en 21 jours »... J'avoue que si j'avais trouvé la même chose pour Gambas, assurément, j'en eusse fait l'acquisition.
Gambas est donc le langage de programmation qui sied avec Linux, et j'essaie de le découvrir en étant sous distribution Ubuntu, en remplacement de VB.
J'ai eu l'idée de commencer cet apprentissage pas à pas, et de noter au fur et à mesure, ce que je pouvais découvrir.
Ceci ne sera donc pas un mode d'emploi pour réaliser telle ou telle application, il n'y aura pas de code à recopier ni de capture d'écran.
Ce programme ne servira à rien, sinon ... à apprendre !

Pré-requis :

On suppose que les personnes susceptibles de me lire ont un minimum de connaissances en programmation. L'environnement de Gambas est assez intuitif et ressemble d'assez près à celui de VB, ou même de Delphi. Tout un chacun sait, comment lancer Gambas, quelles sont ses principales fenêtres, ce qu'est un contrôle et qu'il possède des propriétés et que ces propriétés sont accessibles directement dans la fenêtre de droite lors de la conception, accessibles aussi par le code lors de l'exécution. Par ailleurs, nommer les contrôles pour ce qu'ils sont est absolument nécessaire. Il faut bien savoir que « buton1 » n'est guère plus parlant que « bouton75 », mais que « cmdQuitter », tout comme « txtEntreeAge » indiquent clairement que le premier est le bouton dédié à la fermeture et à l'arrêt du programme, et que le second et une « textbox » qui attend un caractère chiffre, probablement affecté à une durée de vie...
Pour les variables, leur déclaration est obligatoire – ce qui est une bonne chose, et le plus pratique est encore de les nommer d'une façon telle que leur simple nom indique déjà à qui l'on a à faire, tout comme pour les contrôles. Naturellement, quand on arrive à un programme d'importance, il est judicieux de tenir un registre à jour des variables et de leur type de déclaration. Ce genre de document est facile à faire avec un simple fichier.txt et quand il est fait au fur et à mesure de la programmation, cela ne perd pas de temps mais peut en faire gagner beaucoup par la suite. En programmation, l'ordre et la rigueur sont impératifs. Il est d'ailleurs indispensable de commencer ce recensement avant toute écriture de code. Comme sous Linux, il y a plusieurs « bureaux », rien

n'empêche de laisser un fichier.txt dédié, ouvert sur un bureau différent de celui où œuvre Gambas (Au besoin, on rajoute un bureau...).
Le « programme » qui va suivre sera toujours le même, je me contenterai d'ajouts au fur et à mesure que je les découvrirai. Il va s'appeler « Apprentissage000 », et tout au long de cet apprentissage, les sauvegardes seront de 001 puis 002, etc. Ce système est fort pratique car, si (gros) problème il y a, on peut toujours revenir à la version précédente, et repartir d'un endroit resté sain. Cela dit, quand on arrive avec une dizaine de sauvegardes, on peut oser se risquer à supprimer les premières sans pour autant être taxé de témérité.
L'expérience ici mentionnée a été menée avec Gambas 2.13, sous Ubuntu 9.10. Il est probable que les prochaines évolutions modifieront quelque peu le code exposé.

Chapitre 1

On lance Gambas et on choisit « Nouveau projet » et « Application Graphique » et « Suivant ». J'ai créé un répertoire « Gambasse » dans mon Home, mais personne n'est obligé de faire la même chose ! Une fois le répertoire choisi, « Suivant », et là, il faut rentrer, et le nom, et le titre du projet. N'étant pas un adepte de la complication, je rentre « Apprentissage000 » dans le nom et « Apprentissage » dans le titre (au passage, je ne vois pas trop l'intérêt du titre). Ensuite, OK et le projet est parti. Si vous regardez dans votre répertoire Gambasse, vous y trouverez un sous-répertoire « Apprentissage000 », et ce répertoire contient déjà deux fichiers : « FMain.class » et « FMain. Form ».
Revenons à Gambas : On trouve à gauche la liste des éléments qui constituent le programme Apprentissage. Un clic sur FMain et nous avons le premier support graphique du programme. Pour l'instant, ce n'est qu'un rectangle blanc. Si vous « lancez » le programme par F5 (ou par le triangle pointe à droite dans la barre des outils), vous obtiendrez une simple fenêtre avec ses 3 commandes de base (réduit – plein écran – arrêt) et c'est tout ! Si vous ne voyez rien, c'est que la fenêtre est cachée par le programme, il faut la récupérer par un clic, elle doit apparaître en bas, dans la barre d'état (explication et remède un peu plus loin).
Comme la première chose dont on se sert, c'est d'un bouton, on va commencer par en mettre un sur la feuille, sur la « Form ». Si la boîte à outils n'est pas affichée, le mieux est de le faire par clic sur l'icône dans la barre d'outils (promenez la souris au-dessus des icônes pour voir leurs rôles). La boîte étant en bas, à droite, il faut repérer le bouton – un rectangle marqué OK et cliquer, garder enfoncé, et amener le bouton sur la feuille, et relâcher à l'endroit désiré. Le bouton apparaît comme « button1 ». On peut le déplacer, l'agrandir avec la souris. Mais d'abord, on va lui donner un nom : « cmdQuiter ». « cmd » parce que c'est un bouton de commande, ensuite, « Quitter » car il va servir à arrêter le programme. Donc, le bouton étant activé – il a ses 8 poignées en évidence, la fenêtre « Propriétés », à droite, contient toutes les propriétés de ce composant. Pour le nom (name), on va entrer « cmdQuitter », pour le texte (text – ici, ce n'est plus « caption » quoique...), on va entrer « Quitter ». Comme il est un peu réduit sur la feuille on l'agrandit à l'aide de la souris, à partir d'une des poignées. Le texte sur le bouton est bien devenu « Quitter ». Si on lance le programme, la fenêtre apparaît avec son beau bouton, mais si on clique dessus, il ne se passe rien, donc, il va falloir ajouter le code qui va avec. Donc, arrêt du programme soit par clic sur la case système de la feuille, soit par clic sur le carré à droite du triangle de lancement, dans la barre d'outils de Gambas, et retour en mode édition.
Un double-clic sur notre bouton, et là, ô miracle, une autre fenêtre s'ouvre... Avec en prime, un second onglet au-dessus : le premier onglet correspond à la partie graphique (qui vient d'être

remplacée – mais elle est toujours là), le second onglet correspond au code. Ceci n'est d'ailleurs pas sans rappeler les deux fichiers qui se trouvent dans le sous-répertoire de Gambasse, n'est-ce pas ?...
Donc, et dans cette fenêtre, on trouve à peu près ceci :

```
' Gambas class file
' Ceci est un commentaire, il sera ignoré par le code
'Il est uniquement destiné au programmeur pour lui donner des repères
PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()

END

PUBLIC SUB cmdQuitter_Click()


END
```

Bien ! La chose intéressante, c'est la « routine » (ou la Sub...)
Public Sub cmdQuitter_Click().
Une Sub commence toujours par Public OU Private, et se termine par End, et c'est donc entre ces deux lignes que l'on va mettre le code.
Pour fermer « propre » l'application, on met une fermeture de la feuille, et ensuite, on quitte le programme. On va donc trouver ceci pour fermer et arrêter le programme :

```
PUBLIC SUB cmdQuitter_Click()
  ME.Close
  QUIT
END
```

Si on lance le programme, la fenêtre apparaît normalement, un clic sur le bouton « Quitter », et la fenêtre se ferme, le programme se termine, on revient en mode édition.

Voilà pour ce premier contrôle : tous les autres suivront le même cheminement, le même mode opératoire.
Bien entendu, et si cela n'a pas déjà été fait, il faut enregistrer le projet (icône dans la barre d'outils ou par le menu « Fichier / Enregistrer le projet ».
A noter également dans le code, les lignes précédées de ' *sont des commentaires*. Elles servent juste au programmeur pour noter ses remarques, le système les ignore totalement. Il ne faut pas hésiter à en mettre, cela permet de s' y retrouver, surtout lors de la maintenance d'un programme, car il est évident que reprendre un programme plusieurs mois plus tard, ce n'est pas forcément évident.

Chapitre 2

Pour jouer un peu, on va mettre plein de contrôles sur la feuille !
D'abord, une « TextBox » pour entrer des données, numériques ou alphanumériques, et juste à côté, à gauche, un « Label » qui va préciser l'utilité de cette textbox. Ensuite, on va mettre un bouton pour afficher ailleurs – dans un autre label, ce qui a été entré, tout en vidant cette boîte de saisie.
Il faut donc placer sur la feuille une textBox qu'on va appeler « txtSaisie », et renseigner le nom dans ses propriétés . Par la même occasion, il faut aussi effacer le texte « TextBox1 » dans sa propriété « txt », de manière à la présenter vide sur la feuille.
Ensuite, juste à côté, de la « txtSaisie », placer le label qu'on va appeler « lblSaisie », et où l'on va écrire « Saisie » dans la case text de ses propriétés .
Ensuite, on va poser un bouton « Afficher » - nommé « cmdAfficher » et « Afficher » sur sa propriété text, pour faire l'opération.
Enfin, un autre label, appelé « lblAffichage » permettra d'afficher ce qui a été entré dans la boîte de saisie. Une fois son nom entré dans les propriétés, il est utile d'effacer le text par défaut, ici « label1 » si le nom du premier – lblSaisie, a bien été modifié (sinon, ce sera Label2 !).
Tous ces éléments sont à placer au gré de la fantaisie du moment, sur la feuille « Form ».
Pour rappel : la fenêtre des propriétés (à droite) correspond toujours au contrôle sélectionné, même si c'est la feuille. La sélection se fait par un simple clic sur le contrôle souhaité.
A noter que tous ces éléments, y compris la feuille, sont modifiables en taille à tout moment (hors exécution, bien sûr), à l'aide de la souris en cliquant et maintenant les poignées de l'élément sélectionné. Une fois tout ça mis en place, on va voir le code.

Le code va consister à « récupérer » le texte tapé dans txtSaisie et à l'afficher dans lblAffichage.
Comme l'opération va se faire par la commande du bouton cmdAfficher, on va donc s'intéresser à ce seul bouton. Un double-clic cette fois sur ce bouton, et on arrive sur la fenêtre de code,
Il faut bien noter dès maintenant qu'un double-clic sur un contrôle va afficher la Sub correspondante dans la FMAin.Class. Si elle n'existe pas, la Sub sera créée. Si la fonction ne convient pas, on en affectera une autre.
On retrouve la Sub suivante :

```
PUBLIC SUB cmdAfficher_Click()


END
```

C'est donc là que l'on va mettre notre code. Comment faire ?
D'abord, récupérer ce qu'il y a dans txtSaisie Pour cela, on va placer l'instruction :
`Message = txtSaisie.txt`
Ensuite, on va écrire dans lblAffichage avec l'instruction :
`lblAffichage.text = Message`
En réalité, on peut tout faire d'un seul coup en groupant les deux instructions :
`lblAffichage.text = txtSaisie.txt`
Voilà, quand le bouton Afficher va être enfoncé (par un clic), il va porter le contenu de txtSaisie dans lblAffichage !

Reste à vider la saisie, facile :
txtSaisie.txt = ""

Voilà, ça fonctionne ! Un souci cependant : si on clique à nouveau sur le bouton Afficher, il va afficher … la saisie vide... Et il n'y aura plus d'affichage !
La solution va donc consister à bloquer le bouton, tant que la saisie est vide, et le remettre opérationnel dès que la saisie sera renseignée, même avec un seul caractère.
Pour ce faire, on va aller dans les propriétés du bouton Afficher, et mettre à False sa propriété « Enabled ». Quand cette propriété est à « False », le contrôle est bloqué, il faut la mettre à « True » pour que le contrôle soit accessible. Remarquez que quand il est à False, le contrôle apparaît en grisé (valable pour tous les contrôles).
Donc, là, notre bouton est à « Enabled = False », donc il va falloir l'activer dès qu'une saisie aura lieu.
Dès qu'un caractère est entré dans la « txtSaisie », la Sub correspondante est appelée.
Retour sur la feuille graphique et double-clic sur la txtSaisie, le feuille code apparaît, et on trouve ceci :

```
PUBLIC SUB txtSaisie_KeyPress()

END
```

Il suffit de réhabiliter le bouton dans cette Sub, puisqu'elle est appelée à chaque saisie soit :
cmdAfficher.Enabled = TRUE
Mais par ailleurs, il faut aussi le rebloquer dès l'affichage effectué, sinon on se retrouve avec l'effaçage précédent. Pour ce faire, il suffit de rajouter l'opération inverse dans la Sub du bouton d'affichage, ce qui aura pour effet de l'inhiber, dès qu'il aura été sollicité. On va donc mettre dans la Sub du bouton :
cmdAfficher.Enabled = False
Bien, ça fonctionne, mais l'affichage reste un peu discret. Pour le mettre en valeur, il faut aller dans les propriétés du contrôle lblAffichage (un simple clic dessus en mode édition) ensuite, on peut modifier la police par la propriété « Font ». Certes, la case, en face, est vide, un clic et 3 points apparaissent. Un clic sur les points et on a la boîte de dialogue Fonts que tout le monde connaît. Il reste à faire son choix. On peut aussi centrer le message affiché, il suffit de renseigner la propriété « Alignment », et de la mettre sur « Center » par exemple. Si on veut mettre un peu de couleur dans notre contrôle, on va renseigner la propriété « Backgrond », et si l'on craint que le texte soit trop long, ou que l'on souhaite que la case corresponde au texte, on va mettre la propriété « Autoresize » à True. De même, pour mettre un infobulle sur un contrôle, il suffit de renseigner le texte de l'infobulle dans la propriété « ToolTip » du contrôle. Tout ça est à expérimenter … maintenant !

Un petit ajout pour améliorer la présentation :
Comme dit plus haut, il peut arriver qu'au lancement du programme, la feuille n'apparaisse pas et qu'il faille la récupérer dans la barre des tâches, en bas. J'ai eu ce souci, et d'autres, tout cela s'est positionné correctement en allant dans les « propriétés » du projet.
Ces propriétés sont accessibles par la cinquième icône en partant de gauche, dans le menu « Outils », juste après « Enregistrer le projet sous ». Un clic et on trouve une fenêtre : premier onglet, il permet de décrire le projet et d'enregistrer sa version. Second onglet, pas grand chose à toucher, troisième onglet, c'est là qu'il faut intervenir.

Pour ma part, j'ai coché 4 composants : gb, gb.form, gb.qt, gb.qt.ext, et j'ai laissé tous les autres décochés pour l'instant.
Ensuite, il est toujours plus agréable de voir la feuille s'afficher au milieu de l'écran, donc :

ME.Center

dans la PUBLIC SUB Form_Open().

Il faut bien garder à l'esprit que tous les contrôles – et la feuille en est un, ont leurs propriétés accessibles par leur nom auquel on ajoute un point, puis la propriété à modifier. Exemples :
Contrôle.Propriété = valeur, autrement dit :
cmdAfficher.Enabled = True (par exemple)
Me.Center n'a pas de valeur, il se suffit. (Ici, Me représente la feuille)

Normalement et à ce stade, votre code devrait ressembler à ceci :

```
' Gambas class file
PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()
  ME.Center
END

PUBLIC SUB cmdQuitter_Click()
  ME.Close
  QUIT
END

PUBLIC SUB cmdAfficher_Click()
  lblAffichage.Text = txtSaisie.Text
  txtSaisie.Text = ""
  cmdAfficher.Enabled = FALSE
END

PUBLIC SUB txtSaisie_KeyPress()
  cmdAfficher.Enabled = TRUE
END
```

Chapitre 3

Bien, tout ça est bien joli, mais ça ne donne pas grand chose.
Nous allons étoffer le sujet en entrant des données spécifiques. Dans un premier temps, on va entrer le nom du capitaine, et dans un second temps, l'âge du capitaine... Pour l'âge, on va dire que si c'est avant 30 ans, c'est qu'il y a eu magouille, voire népotisme, donc ce sera refusé. Même chose si c'est plus de 55 ans, là, c'est trop âgé, place aux jeunes... Enfin, et pour le nom, on se limitera à 25 lettres, un maximum qui devrait convenir, même aux noms composés qui furent en vogue dans la noblesse d'antan.
Il va falloir changer le nom de la « txtSaisie », que l'on va transformer en « txtSaisieNom ».
 Attention : le fait de changer le nom dans ses propriétés ne va pas changer le nom dans le code !
Donc, et avant de supprimer ou de modifier un contrôle sur une feuille, le mieux est encore de chercher dans le code, toutes les occurrences relatives à ce code et de les supprimer ou des les mettre hors jeu avec des apostrophes pour les rendre comme des commentaires (ce qui permet de les recopier après, en cas de besoin). L'outil « Rechercher » de Gambas est prévu à cet effet. Pour ici, il y a peu de code, le mieux est encore de la supprimer. Il faut faire très attention car un oubli, et c'est le message d'erreur (parfois difficile à retrouver). Donc, on change aussi le nom du label, histoire de se faire plaisir, et on vérifie que tout fonctionne comme avant, en entrant un nom.
Un nom n'est, normalement, pas composé de chiffres. Il faut donc vérifier que la saisie est correcte. Il existe une fonction pour contrôler cela, c'est « isLetter ». Ainsi, l'expression « isLetter(Gambas) » est un booléen, et qui va donner une valeur True, « isLetter(Gambas2) », par contre, va donner une valeur fausse car il y a un chiffre. Si donc, on met entre les parenthèses le contenu de la txtSaisie.text, on aura une valeur vraie si il n'y a que des lettres, une valeur fausse, si il y a un ou plusieurs chiffres. La formule sera :

```
IsLetter(txtSaisieNom.Text)
```

Il suffit donc de mettre la formule dans un If / Then, pour faire ou ne pas faire quelque chose.
Par ailleurs, la commande « txtSaisieNom_KeyPress » montre quelques inconvénients, on va la garder pour autre chose. Pour la circonstance, on va utiliser une autre commande, la commande « Change » soit ici : txtSaisieNom_Change. On va donc créer une nouvelle Sub :

```
PUBLIC SUB txtSaisieNom_Change()
```

A chaque nouvelle saisie dans la txtSaisieNom, la Sub sera appelée et la saisie pourra être récupérée.
Il y a un souci avec « isLetter ». En effet, si le capitaine s'appelle « Dupont-Lajoie », un titre de vielle noblesse, il y aura une erreur... Car cette commande ne reconnaît que les lettres, pas les signes. La solution ne peut se trouver que par un compromis, en l'occurrence, faire en sorte de refuser tout caractère qui se situe avant le code 65, puisque, Scr(A) = 65 (code ascii de A). Malheureusement, on n'est pratiquement dans le même cas de figure qu'avec Isletter, et le tiret et l'espace sont absolument nécessaires dans les noms composés. Une solution possible consistera à refuser l'espace de 1 à 64, mais comme le tiret(-) est à 45 et l'espace à 32, il faudra deux lignes, une de 33 à 44, l'autre de 46 à 64. Bien entendu, si on veut jouer les perfectionnistes, on peut encore en rajouter une pour le 13 (Retrurn), de 14 à 31, mais cela semble peu nécessaire.
Pour les commodités du processus, on peut mettre la valeur saisie sous le nom de « Nom », une variable de type String qu'il faut bien entendu déclarer en début de feuille, puisqu'elle sera utilisée

dans plusieurs endroits, et dans la même idée, une variable « Erreur », de type booléen, également déclarée en début. Ainsi, l'écriture du code sera facilitée.
Maintenant, cliquer sur un bouton pour valider la saisie, ce n'est pas très pratique, et il est plus facile, étant au clavier, de frapper « Entrée / Return », pour valider. Cette fois, on utilisera la Sub « txtSaisieNom_KeyPress », d'abord, pour vérifier que la saisie ne comporte pas de chiffre, et ensuite, pour valider si tout est correct, pour tout effacer et inviter à recommencer si la saisie est incorrecte. Et si la saisie et incorrecte, le mieux est de le dire ! A cet effet, on va créer un nouveau Label, en dessous de la txtSaisie, mais sans fioritures ni bords de manière qu'il soit invisible si rien n'est écrit dedans. Le nom par défaut, ici Label1, peut être conservé, ce contrôle ne devrait pas être beaucoup sollicité dans le code (logiquement, il devrait être nommé).
Si donc après la frappe de Return, le Nom est correct, ou pourra l'afficher, il suffira alors d'utiliser la commande « cmdAfficher_Click » dans le code, l'appeler donne le même résultat qu'un clic sur le bouton, à la souris.
Si l'affichage se fait, il faudra alors donner la main à la saisie de l'âge mais la txtBox requise n'est pas encore créée !
Si le nom est incorrect, il faudra effacer le contenu de la txtSaisie, la remettre en attente de saisie en lui redonnant le **focus**, mettre à néant la valeur de Nom, et à False la valeur de « Erreur » et afficher le message d'erreur dans la Label1 justement créé à cet effet. Tout cela se fait simplement à l'aide d'un « If / Then » en testant la valeur de « Erreur ».


La fonction If / Then se termine normalement par EndIf. Toutefois, si elle n'a qu'une seule issue, on peut tout mettre sur la même ligne et se dispenser du EndIf.

Concernant la saisie de l'âge, on va bien entendu utiliser le même procédé, quoique un peu plus simple pour le contrôle. En effet, la commande similaire, « IsNumber », est ici très efficace et sera utilisée. Enfin, il faudra limiter les entrées comme dit plus haut : moins de 56 et plus de 29. On va donc mettre sur la feuille une nouvelle txtBox, que l'on va nommer « txtSaisieAge », et un label devant que l'on va appeler « lblAge » (mais là encore, on peut se contenter du nom par défaut, ce label ne sera jamais sollicité par le code). Mais il faut d'abord faire un test.
Les Sub nécessaires devraient avoir l'air de ceci :

```
PUBLIC SUB txtSaisieAge_Change()
  'Vérifier que ce ne sont que des chiffres
  IF IsNumber(Val(txtSaisieAge.Text)) THEN Age = Val(txtSaisieAge.Text)
  ' Vérifier l'âge
  IF Age > 29 AND Age < 56 THEN cmdAfficher.Enabled = TRUE
END

PUBLIC SUB txtSaisieAge_KeyPress()
  IF Age > 29 AND Age < 56 THEN
    ' on récupère le code de la touche pressée "Key.Code"
    ' que l'on compare à la constante "Key.RETURN"
    PRINT Age 'Contrôle
    PRINT Key.Return 'Contrôle
    PRINT Key.Code 'Contrôle
    IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
      PRINT Key.Code 'Contrôle
      PRINT Key.Return 'Contrôle
```

```
    cmdAfficher_Click
   ENDIF
  ENDIF
END
```

Voilà, cela fonctionne. Ici, la commande PRINT ne sert qu'à afficher une valeur dans le terminal, la fenêtre en bas, sous le code. On peut l'utiliser pour vérifier une valeur, les lignes concernées seront effacées une fois le contrôle effectué et le programme stable.
Cependant, on s'aperçoit finalement que le bouton « Afficher » est pratiquement devenu inutile, les frappes de « Retrurn » étant suffisantes. On a alors la possibilité d'aller dans ses propriétés, et de mettre sa propriété « Visible » à False. Ainsi, il ne sera plus visible, donc, plus cliquable, mais il fonctionnera pourtant toujours par le code ! Mais bon, pourquoi le garder et encombrer la feuille ? Autant le supprimer sur la feuille directement. Mais attention, sa Sub, elle, est toujours bien en place dans le code. Et bien, il suffit de la renommer en commande. Ainsi, on pourrait l'appeler « Affichage », tout simplement, il y a juste le nom à remplacer. Et comme il n'y a pas de commande, donc, ni « _Click », ni autre, c'est simple...
A noter que les Sub des contrôles doivent être « Public Sub », celles de commande peuvent être « Private Sub ». Finalement, et après vérifications, votre code devrait ressembler à ceci :

```
' Gambas class file
PRIVATE Age AS Integer
PRIVATE Nom AS String
PRIVATE Erreur AS Boolean

PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()
  ME.Center
  ME.SetFocus
  txtSaisieNom.SetFocus
  txtSaisieNom.Text = ""
  txtSaisieAge.Text = ""
END

PUBLIC SUB txtSaisieNom_Change()
  Nom = txtSaisieNom.Text
  IF Len(Nom) > 0 THEN
    Label1.Text = ""
    lblAffichage.Text = "Nom : "
  ENDIF
END

PUBLIC SUB txtSaisieNom_KeyPress()
  ' on récupère le code de la touche pressée "Key.Code" que l'on compare à la constante "Key.RETURN"
  'Si il n'y a pas d'erreur, on affiche
```

```
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    VerifNom
    IF Erreur = FALSE THEN Affichage 'ex cmdAfficher_Click
  ENDIF
  IF Erreur = TRUE THEN
    Label1.Text = "Veuillez entrer un nom correct, SVP..."
    txtSaisieNom.Text = ""
    Nom = ""
    Erreur = FALSE
    txtSaisieNom.SetFocus ' Pour ramener le curseur dans txtSaisie
  ENDIF
END

PRIVATE SUB VerifNom()
  'Vérifier que le nom ne porte que des lettres
  DIM N AS Integer
  FOR N = 1 TO Len(nom)
    IF Asc(Mid$(Nom, N)) > 45 AND Asc(Mid$(Nom, N)) < 64 THEN Erreur = TRUE
    IF Asc(Mid$(Nom, N)) > 32 AND Asc(Mid$(Nom, N)) < 45 THEN Erreur = TRUE
  NEXT
END

PUBLIC SUB txtSaisieAge_Change()
  'Vérifier que ce ne sont que des chiffres
  IF IsNumber(Val(txtSaisieAge.Text)) THEN Age = Val(txtSaisieAge.Text)
END

PUBLIC SUB txtSaisieAge_KeyPress()
    ' on récupère le code de la touche pressée "Key.Code" que l'on compare à la constante "Key.RETURN"
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    IF Age > 29 AND Age < 56 THEN 'on affiche
      Affichage
      Label1.Text = ""
      ELSE ' Erreur, on recommence la saisie !
      Label1.Text = "Veuillez entrer un nombre correct, SVP..."
      txtSaisieAge.Text = ""
      txtSaisieAge.SetFocus
    ENDIF
  ENDIF
END

PRIVATE SUB Affichage() 'ex cmdAfficher_Click()
  IF Age < 30 THEN lblAffichage.Text = "Nom : " & Nom 'txtSaisieNom.Text
  IF Age > 29 THEN
    lblAffichage.Text = "Nom : " & Nom & " - Age : " & Age & " ans"  'txtSaisieNom.Text & " -
Age : " & Age & " ans" 'Attention, tout EST sur la même ligne !
    txtSaisieNom.Text = ""
```

```
    txtsaisieage.text = ""
    Age = 0
    Nom = ""
    txtSaisieNom.SetFocus
    ELSE
    txtSaisieAge.SetFocus
  ENDIF
END

PUBLIC SUB cmdQuitter_Click()
  ME.Close
  QUIT
END
```

Bien, on a vu comment mettre des contrôles sur une feuille, et les lignes de codes qui vont avec. Tous les contrôles obéissent aux mêmes règles, seuls changent les paramètres.
Pour se familiariser avec les contrôles, on va encore ajouter 3 boutons radio pour changer la couleur de fond de l'affichage et 3 autres pour changer la police (Normal, Gras et Italique).
Les « boutons radio » fonctionnent par priorité : celui qui est enfoncé annule les autres du même groupe.

Et justement, il faut faire très attention de bien distinguer les groupes quand on veut une série de boutons radio qui fasse une chose, et une autre série qui fasse autre chose : car tous les boutons sur un même support sont dépendants. A cet effet, on utilise un support, une « Frame », laquelle supporte tous les boutons relatifs à un choix, et cette frame porte aussi un titre qui indique clairement l'objet des boutons. Ainsi, et dans l'exemple ci-dessus, on portera sur la feuille, deux frames, un première, appelée « Fond » et qui permettra de supporter les 3 boutons pour modifier la couleur de fond du label Affichage, et une seconde frame, appelée « Police » et qui permettra de changer l'affichage en normal, gras ou italique.
Un double click sur le bouton 1, appelé ici « rdFond1 » et on a une Sub du genre :
PUBLIC SUB Fond_Click()
qui concernera les boutons « rdFond1 », « rdFond2 », « rdFond3 ».
Même chose bien sûr pour le bouton « rdPolice1 » etc. de l'autre frame. Reste à placer le code.
Pour le fond, c'est relativement simple. Seulement, il faut placer la valeur en hexadécimal. Un solution (pas forcément la meilleure) consiste à se rendre dans la fenêtre des propriétés, au niveau du contrôle « Background », et de choisir la couleur dans la fenêtre de choix qui s'affiche. Il suffit alors de sélectionner et de copier la valeur qui s'affiche, et de la coller dans le code, au bon endroit.
On procède successivement à toutes les couleurs utiliser, les 3 dans ce cas précis. Ainsi, pour modifier la couleur la formule est :
lblAffichage.Background = &HFFFF5F& ce qui aura pour effet de mettre un fond jaune.
Pour les polices, c'est un peu plus compliqué, pour la raison qu'il y a plusieurs critères : nom, taille, gras, italique, souligné, etc, donc, la formule pour passer tout ça est un peu plus complexe. En fait, et si on veut tout modifier, on aura une chose du genre :
lblAffichage.Font = Font["Arial,Bold,Italic,24"]
Si on ne veut modifier qu'un seul paramètre – par exemple, mettre en gras :
lblAffichage.Font = Font["Bold"]

Enfin, pour renseigner le code pour les 6 possibilités, on peut utiliser des « IF », sachant que sur une

seule ligne, le « EndIf » n'est pas obligatoire.
Pour cette partie des boutons radio, votre code devrait ressembler à ceci :

```
PUBLIC SUB Fond_Click()
  IF rdFond1.Value = TRUE THEN lblAffichage.Background = &HFFFF5F&
  IF rdFond2.Value = TRUE THEN lblAffichage.Background = &H00FFBF7F&
  IF rdFond3.Value = TRUE THEN lblAffichage.Background = &H009FFFCF&
END

PUBLIC SUB Police_Click()
  IF rdPolice1.Value = TRUE THEN lblAffichage.Font = Font["Normal"]
  IF rdPolice2.Value = TRUE THEN lblAffichage.Font = Font["Bold"]
  IF rdPolice3.Value = TRUE THEN lblAffichage.Font = Font["Italic"]
END
```

Voilà pour ce complément.

Chapitre 4

« « ***Veuillez retrouver ces lieux dans l'état où vous les avez laissés en partant*** » »
Le programme « Apprentissage » est totalement démuni de mémoire ! En effet, et dès son lancement, toutes les fonctions sont « par défaut », et les boutons radio sont tous inactifs.
Les lignes suivantes vont permettre de mettre en mémoire différentes choses et qui, dès le lancement de l'application, vont s'afficher normalement. En clair, on retrouvera au lancement, l'apparence qu'il y avait à la fermeture, en l'occurrence ici, la couleur et la police utilisées dans la case d'affichage, ainsi que la position des boutons radio. Histoire de s'amuser, on pourra aussi retrouver les nom et âge de la dernière saisie, mais comme ce n'est pas obligatoire, on va mettre un bouton pour l'afficher éventuellement.
Il faudra donc prévoir des variables en conséquences et les enregistrer dans un fichier.txt. On pourra donc déclarer comme variables « Couleur » et « Polisse », pour renseigner les boutons radio et les activer au lancement du programme. Ensuite, on va aussi mémoriser le nom et l'âge qui ont été saisis en dernier. Attention toutefois, la variable « Age » est déclarée numérique, et le fichier et un fichier texte, il ne va donc pas (forcément) l'accepter ! (Tout au moins, pour la version actuelle de Gambas). Il faudra donc créer une seconde variable, par exemple « Agee », qui sera de type « String » (texte), pour qu'elle soit acceptée.
Le code devra être remanié. En effet, à chaque clic sur un bouton radio, il faudra que la valeur soit enregistrée, si on veut la garder. Comme il y a deux groupes de 3 boutons, le plus simple consiste à considérer l'ordre (1.2.3), et à mémoriser l'ordre choisi. Ainsi, si la couleur est orange, on mémorisera « Couleur = 2 », et ensuite au lancement, il suffira de mettre (par le code) le deuxième bouton à « Enfoncé » ou « True », ou « Value = 1 », pour qu'il mette la couleur d'affichage correspondante, tout comme si on avait cliqué sur le bouton radio correspondant. On procèdera de même pour la police.

Les données Nom et Age sont effacées juste après l'affichage, et ce, pour permettre une nouvelle saisie. On va donc mettre en mémoire ces données, juste avant de les effacer.

Pour ce qui est de la lecture du fichier, il faudra qu'elle se fasse dès le lancement de l'application, et immédiatement après, que ces données soient prises en considération.
Pour le dernier nom de la séquence précédente, il n'a d'intérêt qu'au lancement, on va donc prévoir un bouton pour un affichage éventuel. Un fois cet affichage fait – si il est demandé, le bouton n'aura plus aucune utilité, il sera invalidé (enable = false), voire rendu invisible (au choix).
Les Sub de choix correspondant aux boutons devront être modifiées, car jusqu'alors, chaque « If » ne traitait qu'une seule possibilité, ce qui nous dispensait du « EndIf ». Avec la variable couleur ou police, on se retrouve avec deux lignes, donc, il faut le « EndIf »obligatoire.
Pour utiliser le fichier, il faudra deux Sub : une pour l'écriture, l'autre pour la lecture.
D'après ce que je viens d'apprendre sur les fichiers, les commandes sont assez simples. D'abord, il faut ouvrir le fichier, ensuite y lire ou y écrire, ensuite le fermer. Ici, il n'y a que quatre données dans ce fichier, il suffit de les traiter dans l'ordre, sans plus.
Pour ouvrir, il faut utiliser la commande « Open », et le nom du fichier.
Pour lire, il faut compléter le ligne ci-dessus par « For Read », pour écrire, il faut la compléter par « For Create ». En fait la commande « Create » crée le fichier si il n'existe pas encore, elle écrase l'ancien si il existe déjà. Comme les quatre données risquent d'avoir été changées, il n'y a aucune crainte à tout écraser.

Attention : il faut aussi prendre soin de déclarer une variable locale (donc, dans la Sub), qui sera le support de fichier, lequel fichier sera le véritable container des données, et sera enregistré dans un répertoire.
Pour l'exemple, j'ai appelé simplement « Datas.txt » le fichier à conserver, et je lui ai fait un répertoire appelé « Datas » également (mais la confusion n'est pas possible) et ce répertoire est sous-répertoire de celui qui contient l'application soit « Apprentissage000 ». Ma variable nécessaire à l'opération sur le fichier a été nommée « FichierData », et déclarée localement dans chaque Sub, sachant qu'une variable déclarée par « Dim » dans une Sub, n'est pas accessible ailleurs. Ensuite, l'écriture se fait par l'instruction « PRINT #FichierData, Donnée », la lecture par « LINE INPUT #FichierData, Donnée ».
Enfin, et dans un cas comme dans l'autre, il faut penser à refermer le fichier, ce qui sera fait avec « CLOSE #FichierData ».
Une fois le programme lancé, le fichier (Datas.txt) va être lu, et les variables Couleur, Polisse, Nom et Agee seront affectées de leur valeur correspondante. Ensuite, il va falloir agir sur les deux boutons radio concernés. Pour se faire, on va créer une autre Sub, que l'on pourra appeler « ChargerData », et qui fera le nécessaire. Pour faire simple, on peut utiliser la commande « Select Case », une première fois pour examiner les trois cas possibles de couleur, avec « Select Case Couleur », et ensuite, une « Select Case Polisse », pour examiner les trois cas de police. Il suffit de mettre à « Vrai » le Case concerné, et cela validera forcément le bon bouton, tout comme si il avait reçu un clic de souris.
Pour les Nom et Age, il suffit de mettre un nouveau bouton, et dans le code, on reprendra l'affichage qui renseigne la label lblAffichage. Ici, pas besoin de transformation, puisque c'est du texte, on pourra utiliser directement « Agee », alors que dans l'autre cas, c'était « Age » qui était utilisé.
Bien, une fois le code écrit, on lance, on vérifie, et on constate que ça marche … très très mal...
Il s'agit là d'un principe que tout programmeur (même débutant...) découvre très vite :
Si il est relativement facile d'obtenir ce que l'on souhaite, il est beaucoup plus difficile de faire en sorte que le programme ne donne pas ce qu'on n'avait pas prévu...
Et c'est souvent ce « deuxièmement » qui prend le plus de temps !

En effet, si on fait les choses bien dans l'ordre, effectivement, ça passe bien, tout est enregistré normalement. Pour peu que l'on change un peu, par exemple, en laissant le champ Nom, vide, on va

enregistrer un Nom … vide... Même souci quand on change Couleur ou Police après l'entrée d'un Nom ou d'un Age, les paramètres ne seront pas enregistrés. Il faut donc revoir la partie enregistrement.
Il semble judicieux d'enregistrer les valeurs après la validation de la saisie Age, laquelle termine le processus. Cela dit, il faut aussi remettre à « nul » les boîtes de saisies, tant pour le nom que pour l'âge, ceci dans la perspective d'une nouvelle entrée. Une possibilité va consister à utiliser une variable supplémentaire, qu'on peut appeler « Nome » et qui prendra la valeur de Nom que si la longueur de la chaîne Nom est supérieure à 2, ainsi, une chaîne vide sera éliminée, et par ailleurs, il est peu probable qu'un nom fasse moins de 2 caractères. Il faudra aussi bloquer le déroulement dans le cas d'une frappe de Return sur une saisie texte vide, dans l'état actuel du programme, elle serait acceptée et enregistrée ! Dans le même sens, il faudra vérifier que Age est bien compris entre les bornes (30 et55), avant de renseigner la variable « Agee ». Ensuite et seulement, on pourra enregistrer. Pour sécuriser le tout, on pourra bloquer la txtBox qui ne sera pas concernée, en effet, quand on saisit un nom, on ne saisit pas un texte, et vice-versa. Enfin, un enregistrement à la fin de chaque Sub Fond_Click et Polisse_Click permettra de « récupérer » les valeurs en cas modifications après saisies Nom et Age.
Pour ce qui est du rappel de la saisie précédente, on a vu qu'il fallait mettre le bouton à Enable = False dès son unique utilisation, cela dit, si on ne l'utilise pas, il va rester accessible : il faut donc le neutraliser, dès qu'on accède à la première case de saisie, celle du Nom. Pour ce qui est de l'affichage de la mémoire, il ne faut pas oublier qu'on affiche ce qu'il y a dans le fichier, donc, ce seront les variables « Nome » et « Agee » qui seront utilisées, et non pas « Nom » et « Age » !
Après plusieurs essais, il semblerait que l'enregistrement du fichier soit nettement plus adéquat à la fermeture de l'application. En effet, les 4 variables sont toujours en mémoire, elles sont modifiées en cas de besoin, il est donc logique de les enregistrer avant de fermer le programme.

Normalement, pour la partie concernée et à ce stade, votre code devrait ressembler à ceci :

```
' Gambas class file
PRIVATE Age AS Integer
PRIVATE Agee AS String
PRIVATE Nom AS String
PRIVATE Nome AS String
PRIVATE Erreur AS Boolean
PRIVATE Couleur AS String
PRIVATE Polisse AS String

PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()
  ME.Center
  LireFichier
  ChargerData
  txtSaisieNom.SetFocus
  txtSaisieNom.Text = ""
```

```
  txtSaisieAge.Text = ""
  txtSaisieAge.Enabled = FALSE
END

PUBLIC SUB txtSaisieNom_Change()
  Nom = txtSaisieNom.Text
  IF Len(Nom) > 0 THEN
    Label1.Text = ""
    lblAffichage.Text = "Saisie en cours... "
    txtSaisieAge.Text = ""
  ENDIF
END

PUBLIC SUB txtSaisieNom_KeyPress()
  ' on récupère le code de la touche pressée "Key.Code" que l'on compare à la constante "Key.RETURN"
  ' ou Key.Code. Si il n'y a pas d'erreur, on affiche
  Dernier.Enabled = FALSE ' Inhiber le bouton d'affichage de la mémoire
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    IF txtSaisieNom.Text = "" THEN Erreur = TRUE
    IF txtSaisieNom.Text <> "" THEN VerifNom
    IF Erreur = FALSE THEN
      Affichage 'ex bouton cmdAfficher_Click
      txtSaisieNom.Enabled = FALSE
      txtSaisieAge.Enabled = TRUE
      txtSaisieAge.SetFocus
    ENDIF
  ENDIF
  IF Erreur = TRUE THEN
    Label1.Text = "Veuillez entrer un nom correct, SVP..."
    txtSaisieNom.Text = ""
    Nom = ""
    Erreur = FALSE
    txtSaisieNom.SetFocus
  ENDIF
END

PRIVATE SUB VerifNom()
  'Vérifier que le nom ne porte que des lettres
  DIM N AS Integer
  FOR N = 1 TO Len(nom)
    IF Asc(Mid$(Nom, N)) > 45 AND Asc(Mid$(Nom, N)) < 64 THEN Erreur = TRUE
    IF Asc(Mid$(Nom, N)) > 32 AND Asc(Mid$(Nom, N)) < 45 THEN Erreur = TRUE
  NEXT
END

PUBLIC SUB txtSaisieAge_Change()
  txtSaisieNom.Enabled = FALSE
```

```
 'Vérifier que ce ne sont que des chiffres
 IF IsNumber(Val(txtSaisieAge.Text)) THEN Age = Val(txtSaisieAge.Text)
END

PUBLIC SUB txtSaisieAge_KeyPress()
   ' on récupère le code de la touche pressée "Key.Code" que l'on compare à la constante "Key.RETURN"
 IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
   IF Age > 29 AND Age < 56 THEN
     Affichage
     Label1.Text = ""
     ELSE
     Label1.Text = "Veuillez entrer un nombre correct, SVP..."
     txtSaisieAge.Text = ""
     txtSaisieAge.SetFocus
   ENDIF
 ENDIF
END

PRIVATE SUB Affichage() 'ex cmdAfficher_Click()
 IF Age < 30 THEN
   lblAffichage.Text = "Nom : " & Nom
   txtSaisieNom.Enabled = FALSE
   txtSaisieAge.Enabled = TRUE
 ENDIF
 IF Age > 29 THEN
   lblAffichage.Text = "Nom : " & Nom & " - Age : " & Age & " ans"  'txtSaisieNom.Text & " -
Age : " & Age & " ans"  'Sur la même ligne !
   'IF Age > 29 AND Age < 56 THEN
   Agee = CStr(Age)
   IF Len(Nom) > 2 THEN Nome = Nom
   txtSaisieNom.Text = ""
   txtSaisieAge.text = ""
   txtSaisieAge.Enabled = FALSE
   txtSaisieNom.Enabled = TRUE
   txtSaisieNom.SetFocus
 ENDIF
END

PUBLIC SUB Fond_Click()
 IF rdFond1.Value = TRUE THEN
   lblAffichage.Background = &HFFFF5F&
   Couleur = 1
 ENDIF
 IF rdFond2.Value = TRUE THEN
   lblAffichage.Background = &H00FFBF7F&
   Couleur = 2
 ENDIF
```

```
  IF rdFond3.Value = TRUE THEN
    lblAffichage.Background = &H009FFFCF&
    Couleur = 3
  ENDIF
END

PUBLIC SUB Police_Click()
  IF rdPolice1.Value = TRUE THEN
    lblAffichage.Font = Font["Normal"]
    Polisse = 1
  ENDIF
  IF rdPolice2.Value = TRUE THEN
    lblAffichage.Font = Font["Bold"]
    Polisse = 2
  ENDIF
  IF rdPolice3.Value = TRUE THEN
    lblAffichage.Font = Font["Italic"]
    Polisse = 3
  ENDIF
END

PUBLIC SUB Dernier_Click()
  lblAffichage.Text = "Nom : " & Nome & " - Age : " & Agee & " ans"
  Dernier.Enabled = FALSE
END

PUBLIC SUB CreerFichier()
  DIM FichierData AS File
  FichierData = OPEN "/home/toto/Gambasse/Apprentissage001/Datas/Datas.txt" FOR CREATE
  PRINT #FichierData, Couleur
  PRINT #FichierData, Polisse
  PRINT #FichierData, Nome
  PRINT #FichierData, Agee
  CLOSE #FichierData
END

PUBLIC SUB LireFichier()
  DIM FichierData AS File
  FichierData = OPEN "/home/toto/Gambasse/Apprentissage001/Datas/Datas.txt" FOR READ
  LINE INPUT #FichierData, Couleur
  LINE INPUT #FichierData, Polisse
  LINE INPUT #FichierData, Nome
  LINE INPUT #FichierData, Agee
  CLOSE #FichierData
 END

PUBLIC SUB ChargerData()
  SELECT CASE Couleur
```

```
        CASE 1
    rdFond1.Value = TRUE
        CASE 2
    rdFond2.Value = TRUE
        CASE 3
    rdFond3.Value = TRUE
  END SELECT
  SELECT CASE Polisse
        CASE 1
    rdPolice1.Value = TRUE
        CASE 2
    rdPolice2.Value = TRUE
        CASE 3
    rdPolice3.Value = TRUE
  END SELECT
END

PUBLIC SUB cmdQuitter_Click()
   IF Len(Nome) > 2 AND Val(Agee) > 29 THEN CreerFichier 'Enregistrement des données
   ME.Close
  QUIT
END
```

Il est maintenant temps de sauvegarder le travail sous un nouveau nom, on pourra utiliser, en toute simplicité, « Apprentissage001 » Mais il y a soucis.....

Chapitre 5

Comme dit plus haut, un programme donne à peu près toujours ce que l'on veut, malheureusement, il donne aussi très souvent ce que l'on ne veut pas : un appui involontaire sur une touche, une saisie malencontreuse, et on se retrouve parfois avec des comportements assez bizarres.... C'est pourquoi il est important d'apprendre très vite l'usage des outils de débogage.
Le premier, que tout le monde connaît, est le point d'arrêt. En effet, celui-ci stoppe le déroulement du programme à son niveau, et attend une décision de l'opérateur. Pour placer un point d'arrêt, il suffit de faire un double-clic en début de ligne, pour l'enlever, c'est la même opération. Attention, le point d'arrêt ne peut être placé que si le programme est hors exécution ! (Plus vrai sur la 2.19)
Le drame dans un mauvais déroulement, c'est souvent une variable qui change de valeur sans qu'on sache pourquoi, c'est ce qui est arrivé ci-dessus ! On repère assez vite la variable suspecte – généralement, c'est elle qui affiche quelque chose d'inadéquat, aussi, et dès le point d'arrêt posé, on peut assez facilement trouver la valeur d'une variable : il suffit de sélectionner la variable à la souris, et la valeur du moment apparaît dans une infobulle. C'est un peu comme en VB, à la différence que pour VB, le simple fait de mettre le curseur au-dessus de la variable était suffisant.
Ensuite, on a deux outils importants, le « pas-à-pas », qui avance d'une ligne, et le « Avancer », qui continue le programme. Ce sont respectivement les 3 triangles et 2 triangles qui sont à côté de l'arrêt, le petit carré vide. Tout à côté, il y a aussi une flèche pointe en haut vert un tiret, et cela donne une avancée du programme jusqu'à la fin de la fonction courante. Il y a aussi une seconde flèche vers la droite avec un tiret au bout, elle permet d'aller jusqu'à la ligne courante (encore faut-il sélectionner une ligne avant !).
Parmi les autres aides possibles, on peut aussi se créer des lignes de code spécifique. Ainsi, en mettant une ligne supplémentaire, on l'a vu dans le code précédent, du style :

```
Print NomDeMaVariable
```

Dès que l'exécution arrivera à cette ligne, la valeur sera inscrite dans la console, en bas (mais l'exécution ne s'arrête pas !). Cela dit, si la chose est demandée pour plusieurs circonstances, il peut très vite y avoir confusion, et dans ces conditions, il vaut mieux compléter par un texte explicite, on pourra donc écrire ceci :

```
Print "Age = " & Age
```

ainsi, on sera mieux renseigné.
Il y a aussi une autre commande à ne pas négliger, d'autant qu'elle peut être utilisée « normalement » dans le code, pour information :

```
Message.info(Valeur)
```

Ce qui revient un peu au même que ci-dessus, à la différence que là, le programme s'arrête, une boite de dialogue est affichée – avec la valeur (Valeur = Age pourra afficher 35 par exemple), et il faut cliquer sur OK pour que le programme continue son exécution. Ceci peut s'avérer intéressant, non seulement pour suivre une valeur, mais aussi dans le cours d'un programme dans le cas où on a besoin de connaître une valeur à un moment donné, sans avoir spécialement besoin de l'afficher sur la feuille. On peut d'ailleurs utiliser la même syntaxe que ci-dessus :

```
Message.info("Age = " & Age)
```

Avec l'aide de ces outils, on peut cerner une valeur et arriver à trouver l'endroit où la mauvaise action s'est produite. Mais cela est parfois difficile. Dans l'exemple ci-dessus, il y avait, suite à une frappe de Return avec une saisie vide, une mémorisation de la variable « Nom », à une valeur nulle, ce qui entrainait un dysfonctionnement dans l'écriture du fichier.

Il y a aussi une possibilité fort utile dans Gambas, c'est la numérotation automatique des lignes du programme. Il faut aller dans Préférences / Editeur. Cela peut s'avérer intéressant en cas de problème car si le programme s'arrête suite à une anomalie (mauvaise déclaration au autre), une fenêtre donne l'erreur et le numéro de ligne. La numérotation peut alors s'avérer bien pratique.
Pour l'endroit où mémoriser les fichiers, après plusieurs essais, il s'est avéré que c'était encore à la fermeture du programme que cette action était la plus judicieuse.

Néanmoins, et malgré toutes les précautions prises, le code actuel présenté ci-dessus laisse parfois apparaître certaines anomalies, cela dit, dans l'état actuel des choses, et comme c'est de l'apprentissage, des investigations supplémentaires ne sont peut-être pas nécessaires !
Puisqu'on en est ici à l'apprentissage, il est important de savoir que de l'aide est proposée dans différentes pages web. Le forum est aussi très utile, et c'est un peu pour soulager les gens qui répondent que j'ai entrepris la rédaction de cette compilation qui décrit mon apprentissage. Au fond, toutes les questions que j'ai pu poser sur le forum, elles se retrouvent ici, sous forme de réponses.
Le code proposé ici n'a pas valeur de référence, il n'est simplement qu'une indication.
C'est la raison pour laquelle ce texte est en PDF, le « copier / coller » ne serviraient à rien, c'est en codant (au clavier !) qu'on essaie de devenir programmeur...
Cependant, il y a des choses qui sont incontournables, ainsi, et si vous voulez utiliser un fichier, il faudra obligatoirement utiliser un code identique, aux variables personnelles près. La variable « FichierData », qui est ici utilisée peut bien entendu être remplacée par une autre, cela dit, comme il faut qu'elle soit déclarée en début de chaque Sub relative (lire ou écrire), le mieux est certainement de s'en fabriquer une personnelle (si on ne veut pas de celle-là...) et de se la prendre systématiquement. Attention aussi pour le chemin : à un moment dans l'exemple, j'ai écrit le nom entier du chemin ce qui, normalement, permet à l'application de tourner ailleurs, en particulier, si l'on fait un exécutable. J'ai lu ci et là que le chemin pouvait être relatif, en particulier, si le répertoire était sous-répertoire de celui où est l'application, j'ai aussi vu une explication où l'on donnait une possibilité de chemin relatif en commençant le nom du chemin par tilde (~), j'ai fait quelques essais, mais je n'ai rien trouvé qui fonctionnait. Il est aussi possible que ma version actuelle de Gambas (2.13), ne permette pas cette fonctionnalité.
(Il faudra voir à installer une version plus récente dans les prochaines semaines, mais pour l'apprentissage basique, celle-ci peut assurément suffire.)
Cependant, il existe une (très) bonne solution :

```
FichierData = OPEN Application.path & "/Datas/Monfichier.txt" FOR [Read - Create]
```

Cette syntaxe fonctionne normalement. En effet, il est normal de mettre des répertoires pour séparer les différents éléments accessoires au programme : données, images, sons, etc., aussi, il est logique de créer des sous-répertoires spécifiques, et directement fils du répertoire qui contient l'application, ainsi, tout est regroupé logiquement et en ordre.
J'ai parlé plus haut d'exécutable. Il est possible de faire un exécutable avec Gambas, il y a une icône prévue à cette effet dans la barre des outils. C'est la neuvième, en partant de la gauche, celle qui a les trois engrenages. On obtient un fichier au nom similaire, mais avec l'extension « .gambas ». Il peut être placé dans un autre répertoire, sous réserve que les chemins des éventuels fichiers soient écrits en totalité ou mieux, que les sous-répertoires concernés soient copiés dans le répertoire de l'exécutable.


Une autre chose aussi importante, c'est la sauvegarde régulière de son travail. Programmer est long et demande des efforts, il est profondément navrant de voir toute une partie de son travail perdue parce qu'il y a eu panne, ou que vous avez créé une boucle infinie, et que votre programme ne vous donne plus la main, et qu'il devient impossible à arrêter. Si vous êtes encore avec la fenêtre d'édition au second plan, ça peut aller, sinon, si vous êtes en mode « plein écran », ce sera impossible. C'est pourquoi, si on souhaite une présentation type plein écran, il faut être absolument sûr de son code !
Je n'ai pas vu de sauvegarde automatique dans les « Préférences ». En tous les cas, il ne faut pas hésiter à sauvegarder très régulièrement son travail.
J'ai été surpris par le bouton « Enregistrer sous ». L'utilisation de ce bouton m'a apporté des ennuis que je n'explique pas. En fait, Gambas donne comme nom à l'application le nom du répertoire qui la contient et donc, normalement, cette commande devrait créer un autre dossier. Or, ce n'est pas le cas, enfin si, mais le dossier créé est ainsi créé dans le dossier même où est déjà l'application. Ce qui fait que la sauvegarde crée un dossier qui crée un dossier qui crée un dossier … jusqu'à remplissage de la pile ! A titre d'exemple, j'ai compté 119 sous-sous-sous.... répertoires !
La solution trouvée, et retenue – pour l'instant, consiste à créer un autre répertoire (avec le gestionnaire de fichiers), et lui donner le nouveau nom, et ensuite, de coller, dans ce répertoire, tous les éléments de l'ancien, y compris les sous-répertoires et leurs fichiers éventuels. Naturellement, il ne faudra pas oublier de modifier les chemins complets dans le code (si ils existent), sinon, ce seront les anciens fichiers qui seront pris en compte (ce qui peut ne pas être un problème si ils sont dans un répertoire spécifique. Il semblerait que cette fonction « Enregistrer sous » soit une lacune de Gambas, qui sera probablement très vite corrigée. Mais dans l'état actuel des choses, sauvegarder sous un nouveau nom pose vraiment un gros problème.

Bien, ces précisions étant apportées, on va pouvoir retourner aux contrôles et au code !

Chapitre 6

Voilà, maintenant que nous avons tous les atouts pour suivre l'évolution d'un programme, on va continuer à découvrir les contrôles.
Taper un nom dans une boite de saisie, puis un âge dans une autre, c'est bien. Mais on peut aussi noter que chaque nouvelle saisie détruit la précédente, et la mémorisation de la dernière entrée n'est assurément qu'un pis-aller. Avec la « ListBox », tout va être conservé ! Tout au moins, tant que le programme sera actif !
Donc et sans plus attendre, retour à la fenêtre FMain.form, là où est le graphisme.
Dans la boîte à outils, et en promenant un peu la souris, on trouve le contrôle ListBox (premier onglet de la boîte). On le tire avec la souris sur la feuille, et on le place là où ça fait le mieux, au besoin, on agrandit la feuille, les poignées sont là pour ça. Au passage, on donne à notre listBox un nom moins anonyme, « lstBoxNoms » par exemple.

Parenthèses : L'usage de préfixes pour les noms de contrôles est purement personnel : mais il a le mérite d'être clair, quand on emploie toujours les mêmes : lst est une liste, cmd, un bouton, rd, un bouton radio, etc. Voir liste en fin.
Voilà, maintenant, retour au code.
La listBox étant installée et nommée (lstBoxNoms), il faut la remplir. Pour mettre une ligne dans la ListBox, il faut utiliser la commande « lstBoxNoms.Add(Chaîne à afficher). Dans le code précédent, tout est déjà prêt : il suffit de retrouver la ligne de code qui affiche le nom ET l'âge dans la lblAffichage, et d'ajouter une ligne supplémentaire quasiment identique. Ce qui devrait donner ceci :

```
lblAffichage.Text = "Nom : " & Nom & " - Age : " & Agee & " ans"
lstBoxNoms.Add("Nom : " & Nom & " - Age : " & Agee & " ans")
```

Et dès qu'une nouvelle entrée sera validée, elle sera affichée, et dans le label, et dans la listBox.
Cela dit, un observateur attentif aura remarqué une légère différence par rapport au code mentionné au chapitre précédent : Compliments !
En effet, si on se contente d'ajouter une seule ligne, tout va bien. Dans le cas de deux saisies successives, la variable « Age » n'était remise à zéro que après l'affichage. Ce qui faisait qu'à partir de la seconde saisie, la valeur « age » était toujours correcte, et en conséquence, acceptée comme telle et engendrait la même variable « Agee » ! Cela confirme bien ce qui est dit plus haut : éviter que le logiciel ne fasse ce que l'on ne veut pas ! Donc, en plaçant la conversion (Agee = Cstr(Age)) avant l'affichage, le problème est résolu.
Donc, et si on tape plusieurs entrées de suite, elles vont s'afficher dans l'ordre, dans la listBox. Dans le code ci-dessus, il y a, non seulement les variables, mais aussi du texte supplémentaire, peut-être pas nécessaire, mais utile (à chacun de juger).
Il est aussi possible que l'on veuille enlever une ligne de la listBox, voire, tout vider...
Facile : Un bouton sur la feuille (cmdSupprimer), un autre bouton (cmdVider), un double-clic sur le premier, et on a la Sub conséquente « PUBLIC SUB cmdSupprimer_Click() », idem sur le second.
Les listBox ont leurs lignes repérées (en interne) par un compteur nommé « Index ». En cliquant sur une ligne, la valeur Index, est immédiatement mise un mémoire, il suffit de l'utiliser pour supprimer la ligne. La commande sera :

```
lstBoxNoms.Remove(lstBoxNoms.Index)  ' Supprimer la ligne
```

Pour vider toute la liste, il y a une commande spécifique :

```
lstBoxNoms.clear   ' Vider la boîte
```

C'est vraiment très simple ! Maintenant, et si on veut se faire plaisir, on pourrait afficher le nombre de lignes rentrées. Quand elles sont peu nombreuses, c'est facile de se repérer, à partir d'une dizaine, c'est moins évident de s'y retrouver. On a vu la propriété « Index », qui est en fait un pointeur de ligne : pourquoi ne pas l'utiliser ? On verra après. Pour l'instant, on a une autre propriété, « Count » qui donne directement le nombre de lignes dans la listBox. Il suffit donc de placer un nouveau label sur la feuille – ici lblNombre, et de placer aux endroits adéquats la ligne suivante :

```
lblNombre.Text = lstBoxNoms.Count
```

Ainsi, le nombre de lignes sera affiché. On pourra mettre cette ligne après la validation d'une nouvelle entrée, après une modification, ligne effacée ou tout effacé.

Attention, avec « Index » : il commence à zéro pour la première ligne, 1 pour la seconde. Pour mettre cela en évidence, créer une nouvelle Sub avec un double-clic sur la listBox : on a alors une Sub, dans laquelle on va ajouter une « Message.info », et qui donnera la valeur de l'index en cliquant sur une ligne de la box. Enfin, il serait bon d'utiliser cette propriété pour « récupérer » le contenu d'une ligne, pour faire simple ici, on va l'afficher, et on va utiliser en même temps la Sub Click, mentionnée ci-dessus.

Une fois « Message.Info » étant vu et bien vu, il n'est peut-être pas utile de le conserver. C'est selon. Pour afficher la ligne sélectionnée, il suffit de rajouter deux lignes dans la Sub Click :

```
lblAffichage.Text = lstBoxNoms.Text
```

Cette méthode permet de récupérer la valeur d'une ligne pour différents usages, en fonction des besoins. On peut aussi vouloir accéder à une ligne à partir de son numéro. Il suffit de rentrer un nombre avec une ValueBox par exemple. On va donc placer une ValueBox sur la feuille, lui mettre un Label devant pour expliquer son usage, et écrire le code correspondant. Attention cependant : ici, on va faire appel à la propriété « Index », et il ne faut pas oublier qu'elle commence à zéro pour la première ligne ! On va donc entrer le numéro de ligne à traiter, lui retirer la valeur 1, et l'affecter à la propriété index. Le simple fait de modifier l'index va activer la ListBox, comme si on avait cliqué sur la ligne considérée, ce qui va l'afficher...

A ce stade, votre code (relatif à la ListBox), devrait ressembler à ceci :

```
PUBLIC SUB cmdSupprimer_Click()
  'Supprimer l'élément sélectionné de la liste
  lstBoxNoms.Remove(lstBoxNoms.Index)
  IF lstBoxNoms.Count < 1 THEN
    cmdSupprimer.Visible = FALSE
    cmdVider.Visible = FALSE
    vlBox.Visible = FALSE
    Label3.Visible = FALSE
  ENDIF
  lblNombre.Text = lstBoxNoms.Count
END

PUBLIC SUB cmdVider_Click()
  lstBoxNoms.clear
  cmdSupprimer.Visible = FALSE
  cmdVider.Visible = FALSE
  vlBox.Visible = FALSE
  Label3.Visible = FALSE
  lblNombre.Text = "0"
```

```
END

PUBLIC SUB lstBoxNoms_Click()
  'Message.Info(" Index = " & lstBoxNoms.Index) ' A utiliser ou non
  lblAffichage.Text = lstBoxNoms.Text
END

PUBLIC SUB vlBox_KeyPress()
  DIM Nbre AS Integer
  Nbre = vlBox.Text - 1
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    'lblAffichage.Text = lstBoxNoms.List[Nbre] ' 1ère méthode : directe
    lstBoxNoms.Index = Nbre        ' 2 ème méthode : va activer lstBoxNoms_Click
  ENDIF
END
```

Il y a deux possibilités pour récupérer une ligne : la première est « directe » et totalement transparente. La seconde active la ligne dans la ListBox. Suivant les besoins, l'une ou l'autre sera utilisée. La première permet de récupérer directement la ligne dans une variable dédiée, la seconde à le mérite d'afficher l'opération en quelque sorte !
La ListBox permet donc de laisser affichés les noms entrés, mais ils sont perdus dès la fermeture du programme. Si on veut les conserver, il faut les enregistrer dans un fichier, et lire le fichier au lancement pour les récupérer, tout comme les éléments précédents qui sont mémorisés dans cet exemple, c'est-à-dire, la couleur de l'affichage, la police, et le dernier entré. Or, avec ce nouveau système, le dernier entré n'a plus de raison d'être. On va donc supprimer cette fonction et la remplacer par une autre. Mais par souci de clarté, on va faire un second fichier pour les listes, et garder l'actuel fichier pour les paramètres. Dans un premier temps, on va supprimer tout le code relatif à la Sub et la Sub elle-même. Il est possible (et même certain) que des éléments restent dans le code, et ils vont faire une erreur à l'exécution. Il faut donc les retirer. Avec l'outil « Rechercher », on peut les localiser facilement, et les enlever. Il faut lancer l'outil « Recherche » (les jumelles), et avec la flèche entre les deux fenêtres de l'outil « Rechercher / Remplacer », on passe d'une occurrence à l'autre, il suffit d'effacer la ligne de code qui porte le mot « Dernier ».
Une fois le ménage fait, on va utiliser le bouton pour enregistrer la liste. Retour dans la Form, on va renommer le bouton « Dernier » en « cmdEnregistrer la liste » (au passage, le bouton aurait du, selon toute logique, être appelé « cmdDernier », c'est un oubli...), donc, on renomme le bouton, on change son texte, et on laisse ses propriétés en l'état. Si le bouton n'est pas assez large, on l'agrandit ! Et si un autre à côté n'est pas à la même cote, on l'agrandit aussi, sinon, ça fait désordre...
Comme on n'enregistre plus la dernière entrée, il faut revoir la fonction enregistrement précédent. Pour l'instant, il n'y a plus que deux choses à considérer : la couleur et la police. Il faut donc changer le code dans les Sub correspondantes. Pendant qu'on y est, on va même changer le nom du fichier. Il faut encore faire attention : en effet, si on change tout d'un coup, au lancement de l'application, la Sub de lecture – puisque c'est elle qui est lue en premier, va trouver l'ancien fichier, et comme les données ne seront plus les mêmes, il y aura erreur. Dans ce cas de figure, le plus simple est encore d'écrire d'abord, la Sub Create, de lancer le programme, ainsi, un nouveau fichier sera créé. Après quoi, on ferme le programme et on écrit la Sub de lecture qui trouvera forcement le nouveau fichier, donc, le fichier adéquat. Il est certainement raisonnable de toujours garder ce mode opératoire avec les modifications de fichier.

On va donc créer une nouvelle Sub qui va nous générer un fichier que l'on va appeler « Parametres ». Pour l'instant, il n'y aura dedans que « Couleur » et « Polisse », mais par la suite, il sera toujours possible d'y ajouter des nouveaux éléments. Le processus sera d'ailleurs toujours le même : modifier la Sub Create, faire tourner le logiciel pour créer le nouveau fichier, puis écrire ensuite le Sub Lecture.


Quand on a l'idée d'un projet, il doit être soigneusement préparé, on ne se lance pas comme ça, tête baissée dans les contrôles et dans le code, il faut beaucoup y réfléchir avant. Il faut garder à l'esprit ce que l'on veut faire, comment on va le faire, quelles routines (Sub) on va utiliser, quels contrôles, quelles variables. Il y a des variables importantes que l'on va devoir solliciter tout au long du programme, d'autres, tout à fait occasionnelles, locales, celles-ci ne méritent pas une réflexion poussée. Cela dit, un peu méthode ne nuit pas et on peut tout à fait décréter que la variable N sera systématiquement déclarée et utilisée dans toutes les Sub qui utiliseront une boucle For Next. C'est au fond le même exemple que pour le nom (provisoire) de fichier dans les accès lecture / écriture, et qui s'appelle ici FichierData. Ce genre d'habitude a le mérite de la clarté et de la simplicité et il fait gagner beaucoup de temps ! Bref, avant de se lancer dans le code, il faut avoir utilisé un crayon et … beaucoup de papier...
Une parenthèse car si un « pro » venait à lire mon texte, il serait horripilé d'entendre parler de Sub ! En effet, il y a deux sortes de Sub, celles qui font une tâche particulière et rien de plus, et on les appelle des « Procédures », soit qu'elles reçoivent une donnée via les parenthèses terminales, et rendent une donnée modifiée, et dans ce cas, on va parler de « Fonction ». Mais comme je ne suis pas un « pro », le mot Sub ne m'agresse pas...

Bien, revenons à notre fichier « Parametres ». Comme il y a déjà une Sub CreerFichier, le mieux est de la modifier, d'autant que peu de choses vont changer. Comme on va en avoir une autre, on va la nommer « CreerFichierParametres ». Ensuite, il faut changer le nom Datas par Parametres dans la ligne Open. Il reste à effacer les deux dernières lignes, et … c'est tout ! Eh non, ce n'est pas tout... Lancez l'application, et vous avez une erreur : en effet, l'appel à la Sub n'a pas été modifié, il est fait depuis la Sub Quitter. On modifie puis on relance, puis on arrête, ensuite, on vérifie avec son gestionnaire de fichiers préféré que le fichier « Parametres » est bien en place, et si on l'ouvre, on constate qu'il comporte les deux numéros de couleur et de police. Maintenant, il faut aussi modifier la Sub Lecture, sans oublier la ligne qui l'appelle... Là aussi, on va renommer par « PUBLIC SUB LireFichierParametres() », modifier le nom et supprimer les deux dernières lignes.
Il est temps d'envisager maintenant l'enregistrement des données affichées dans la ListBox, de manière à les retrouver au lancement du programme. Dans un premier temps, on va faire simple, l'enregistrement complet de la ligne, ce qui fera un affichage tout aussi complet.
Il faut déjà créer une procédure pour écrire ou créer le fichier, fichier que l'on peut appeler « Donnees.txt », sans plus d'originalité. Ce sera donc : PUBLIC SUB CreerFichierDonnees().
Pour ce faire, on va d'abord déclarer un fichier comme dans le cas du fichier « Parametres », et comme il y a plusieurs données à entrer, on va se servir d'une boucle For Next : il faut donc déclarer une variable N qui sera utilisée dans la boucle. Pour documenter la boucle, on se servira judicieusement de la propriété « Count » du label, en ayant soin de lui soustraire la valeur 1, puisque le traitement va se faire par la propriété « Index », et que l'on a vu que le numéro d'index commençait à zéro ! Ensuite, il suffira d'utiliser la propriété de « List » pour renseigner chaque enregistrement.
Une fois la Sub créée, il faudra l'utiliser. On va se servir du bouton resté en attente, l'ancien bouton « Afficher dernière entrée ». Tout son code a été effacé, on va en mettre un autre. Un double-clic sur le bouton, on a une Sub_Click dans laquelle il suffira d'appeler la Sub d'enregistrement.

Pour continuer, on doit envisager d'afficher le contenu du fichier dès le lancement de l'application. D'abord, il faut faire une Sub de lecture. A priori, on ne sait pas, au lancement, combien le fichier compte de lignes. Et tenter de lire une ligne qui n'existe pas, ça génère une erreur. Les fichiers ont une propriété intéressante, c'est un indicateur de « Fin de fichier », et qui se nomme « EoF », il suffit donc de l'utiliser dans une boucle While ! Ensuite, à chaque ligne lue, il faudra ajouter une ligne à notre ListBox, (par la méthode Add) mais comme cela ne peut se faire directement, il faudra prévoir une variable locale, de type String, qu'on peut appeler simplement « Ligne ». Ensuite, et pour que l'action se produise, au lancement, il faut ajouter l'appel de la Sub LireFichierDonnees dans la Sub Form de départ, par exemple, juste après la ligne qui appelle la lecture des paramètres.

A ce stade, le code concernant l'exploitation du fichier des données, devrait ressembler à ceci :

```
PUBLIC SUB cmdEnregistrerListe_Click()
  CreerFichierDonnees
END

PUBLIC SUB CreerFichierDonnees()
  DIM FichierData AS File
  DIM N AS Integer
  FichierData = OPEN Application.path & "/Datas/Donnees.txt" FOR CREATE
  FOR N = 0 TO lstBoxNoms.Count - 1
  PRINT #FichierData, lstBoxNoms.List[N]
  NEXT
  CLOSE #FichierData
 END

 PUBLIC SUB LireFichierDonnees()
  DIM FichierData AS File
  DIM Ligne AS String
  FichierData = OPEN Application.path & "/Datas/Donnees.txt" FOR READ
  WHILE NOT Eof(FichierData)
  LINE INPUT #FichierData, Ligne
  lstBoxNoms.Add(Ligne)
  WEND
  CLOSE #FichierData
 END
```

Voilà, ça fonctionne. Mais on peut aussi mettre l'appel à l'enregistrement (CreerFichierDonnees), dans la Sub Quitter, ainsi, à la fermeture du programme, le fichier sera enregistré systématiquement. Ceci peut être utile, en particulier dans le cas où des données ont été enlevées de la ListBox.
Il est possible de donner le choix à l'utilisateur et de mémoriser ce choix. On peut aussi donner un avertissement en cas de modification de la liste. Mais dans ce cas, il faut aussi prévoir un booléen qui indiquera que des changements ont eu lieu, que se soit en ajout ou en retrait.
Pour mettre en œuvre ces deux possibilités, on va d'abord utiliser une CheckBox pour mémoriser la sauvegarde automatique ou pas, du fichier Donnees.txt, et pour prévenir d'une modification, ce sera une « Message.Warning ».

Donc, retour vers la FMain.form et ajout d'une Checkbox que l'on va aller chercher dans la boîte à outils. On va la nommer « chkSave », et comme il faudra mémoriser sa valeur, on va créer une variable « SaveDonnees », de type booléen, et que l'on va déclarer dans le début du programme. Naturellement, cette variable devra être mémorisée, dans le fichier « Parametres », il suffit de l'ajouter.

Attention à ce qui a été dit plus haut : si une variable est ajoutée à un fichier qui est lu au lancement du programme, cela va générer un erreur, il faut donc ne pas lire le fichier avant qu'il ne soit enregistré avec son nouvel ajout, la procédure a été décrite ci-dessus.
Mais après ajout de cette variable dans les deux Sub (Lire et Créer), on constate que ça ne marche pas. A la lecture, la variable revient systématiquement à True : un mystère à élucider... En fait, le fichier ne semble pas accepter un booléen...
Pour contourner ce petit souci, la solution consiste à créer une variable locale, que l'on va déclarer « DIM Save AS Byte », et on va lui affecter la valeur 1 si notre booléen SaveDonnees est vrai, zéro dans le cas contraire. Bien entendu à la lecture, on fera l'opération inverse.

Pour ce qui est de la variable « Modif », de type booléen, elle n'a pas besoin d'être mémorisée, elle sera elle aussi à placer au début du programme, et False (Fausse) au lancement.
Donc, la checkBox est en place, renommée et son texte sera: « Sauvegarde automatique des données ». Un double click dessus, et on se retrouve avec la Sub adéquate. Attention aussi de renseigner la checkBox dès le début, dans la Sub ChargerData, avec la ligne suivante, de manière à ce qu'elle retrouve son dernier état :

```
IF SaveDonnees = TRUE THEN chkSave.Value = True
IF SaveDonnees = FALSE THEN chkSave.Value = False
```

Nota : on pourrait aussi laisser sa propriété de base « Value » à False et n'utiliser que la première ligne.
Voilà, dans la Sub Quitter, nous avons un If qui va appeler la sauvegarde ou pas, du fichier de données, suivant l'état de le checkBox. La sauvegarde des paramètres reste toujours activée.
Si la sauvegarde automatique n'est pas activée, il faut penser aux gens distraits. A cet effet, on va déclarer un booléen « Modif » au début du programme, et le mettre à True à chaque intervention, c'est à dire, en cas d'ajout, de retrait, ou de vidage de la liste. Si il a la valeur vrai, c'est-à-dire, si une modification a été faite dans la liste, un message d'avertissement va être présenté, avec la possibilité de sauvegarder malgré tout.
A ce stade, on pourrait envisager de supprimer le bouton « Enregistrer la liste ». On peut cependant le conserver, pour une utilisation ponctuelle, par exemple, on peut vouloir enregistrer le fichier à un instant T, avant de le modifier par exemple, puis on en garde une copie par son gestionnaire préféré, après quoi, on peut modifier et enregistrer le fichier modifié, et on garde ainsi une trace de l'ancien.
On peut faire encore mieux : l'utiliser la fonction « Enregistrer sous ». Ainsi, on aura accès à une boîte de dialogue permettant de renommer, et le fichier, et le chemin.
Naturellement, il faudra changer la validité du bouton : Valide si la ListBox contient au moins un élément, Invalide si la ListBox est vide. Ne pas oublier qu'on peut vider la liste par le bouton « Supprimer de la liste »... On va donc mettre la propriété Enabled à False dans les propriétés, la mettre à True au chargement au lancement, seulement si il y a quelque chose dans le ficher (et si il y a fichier...). Ensuite, False avec « Vider la liste », et False si la liste devient vide par « Supprimer ».

Pour la commande « Enregistrer sous », les données ne sont pas dans une ou des variables, mais dans la ListBox. Pour créer et enregistrer un nouveau fichier, il faut d'abord déclarer une variable locale, ici ce sera « Aenregistrer », et la remplir avec les éléments de la ListBox. On sait que chaque

élément peut être extrait avec la méthode List, on sait leur nombre, avec la méthode Count. Avec ces deux éléments, on va faire une boucle pour renseigner la variable, avec, à chaque itération, une concaténation Aenregistrer = Aenregistrer & élément suivant de la liste. Le souci, c'est que, à la première ligne, la variable va être vide et ajoutée, ce qui donnera une première chaîne vide dans le fichier. Pour contourner le problème, on peut assigner la première donnée de la ListBox à notre variable Aenregistrer, et ensuite, continuer dans une boucle.
Attention, et comme déjà dit, ici, c'est la valeur d'index qui gère, donc, et comme la première ligne sera déjà lue, il faudra commencer la boucle à 1 pour lire la ligne 2, la ligne 1 étant déjà renseignée.
Naturellement, comme on connaît le nombre lignes par la propriété Count, on peut tout à fait utiliser une boucle For Next comme on l'a fait dans la SUB CreerFichierDonnees.

A noter encore un détail : Au lancement de l'application, on va aller lire le fichier des paramètres, puis le fichier de données. Seulement, les fichiers peuvent très bien être absents, pour différentes raison, ne serait-ce que l'absence au premier lancement. Il faut donc prévoir une gestion d'erreur. Il faut d'ailleurs toujours prévoir une gestion d'erreurs avec les fichiers !
En Gambas, cela se fait pas le mot-clé CATCH, et généralement, on prévient par une Message.Info(Error.Text) . Error.Text semble avoir ceci de particulier qu'il donne l'erreur directement, mais en anglais... Mais voilà qui renseigne très facilement.
Bien, nous avons une liste de personnes, et la possibilité d'ajouter, d'enlever, de sauvegarder, et aussi d'afficher. Pour l'affichage justement, il pourrait être agréable d'avoir un automatique, un peu comme un diaporama en photos. C'est simple !
Retour sur la feuille, on rajoute une frame. Pas besoin de lui donner de nom, elle ne sera jamais sollicitée par le code. Par contre, son texte devra être explicite, ici on va écrire : « Défilement automatique ». On ajuste tout ça sur la feuille, de manière à faire quelque chose de présentable.
Sur cette frame, on va placer un bouton pour faire démarrer le défilement. On va l'appeler « cmdDefilement », et on va mettre son texte à « Départ ». On va utiliser deux vitesses de défilement, une fixe à 1 seconde, l'autre variable. Pour le choix, on pose deux boutons radio, le premier « rdFixe » aura son text à « Fixe : 1 s », et on va mettre sa propriété Value à True, de manière qu'il soit enfoncé au départ de l'application. Le mode fixe sera donc le mode par défaut.
Pour faire varier la vitesse, on va utiliser un curseur, « slader », contrôle que l'on va trouver dans la boîte à outils, premier onglet, plutôt à gauche. On amène le slader sur la feuille, dans la frame dédiée, on redimensionne au besoin. Ensuite, il va rester à poser un label pour afficher le temps déterminé par le slider : « lblTemps » conviendra très bien pour indiquer de qui de quoi il s'agit. Il est en effet important de bien présenter et de bien renseigner les outils opératoires.
Une fois tout cela posé sur la frame, il faudra sûrement mettre un peu d'ordre pour présenter quelque chose qui ne soit, ni trop fouillis, ni trop moche, ni trop entassé : ici, le « sens esthétique » de l'opérateur est nécessaire (ce que je ne revendique absolument pas...).
Mais pour faire tourner tout ça, il faut une horloge, on va donc mettre sur la feuille, un « Timer ». Ce contrôle se trouve, comme tous les autres, dans la boîte à outils, mais dans le dernier onglet (attention, il est un peu caché, il faut aller le découvrir avec les flèches de la boîte).
Le timer peut être mis n'importe où sur la feuille, il n'apparaît jamais. On va lui donner le doux nom de « TimerDefile » et mettre sa propriété Delay à 1000 soit, mille millisecondes, soit une seconde.
Maintenant, il va falloir s'occuper du code...
On va d'abord s'occuper du bouton : un double-click, et on trouve la Sub toute neuve dans le code.
Ici, on va faire un bouton à bascule : Pour l'instant, il est affiché Départ, si on clique dessus, il devra se passer deux choses : la première, le Timer va démarrer et faire defiler les noms dans lblAffichage, la seconde chose, c'est que notre bouton va changer de fonction, il va devenir « Stop », et si on clique à nouveau dessus, il devra arrêter le processus et revenir à l'affichage Départ. Pour ce faire, il faut utiliser un booléen qui va retenir l'état du moment. On ne peut pas utiliser un booléen

local (déclaré par Dim) car il serait revalidé à chaque clic, il faudra donc l'ajouter aux variables déjà nombreuses qui trônent au début du code. On va donc ajouter « PRIVATE Defil AS Boolean ». Et pour être sûr du processus, on va le déclarer à False dans la Form_Open, (celle qui est lue en premier), ce qui est normal, puisque rien ne défile pour l'instant. Maintenant, on va séparer notre Sub en deux : une première moitié pour Départ, une seconde moitié pour Stop. On va donc écrire ceci :

```
PUBLIC SUB cmdDefilement_Click()
  IF Defil = FALSE THEN 'Pour afficher le défilement

  ENDIF
  IF Defil = TRUE THEN 'Pour arrêter

  ENDIF
END
```

Ensuite, les trois choses à faire seront :

- de changer la valeur du booléen Defil,
- de changer l'affichage du bouton,
- et de lancer le Timer dans la première partie, de l'arrêter dans la seconde.

Le Timer va aller dans sa Sub, et à chaque appel, il va exécuter le code de sa Sub. Pour sa fréquence, elle est déjà programmée à 1000 pour l'instant. Donc, un double-click sur le Timer, et on a la Sub.


Attention quand même : il faut bien avoir à l'esprit que le booléen conditionne le fonctionnement du code bouton. Ainsi, quand on est dans la première partie, celle où l'on veut démarrer, on va mettre le booléen à True, en prévision de Stop, ce qui va faire en sorte que le code, qui lit les lignes les unes à la suite des autres, va trouver la seconde alternative (IF Defil = TRUE THEN) devenue recevable ! Il faut donc prendre soin de mettre une sortie de la Sub en fin du premièrement pour ne pas aller lire la seconde partie. Par contre, pas de souci, si le booléen est à True, la première partie va être ignorée, et tout se passera normalement. Pour sortir d'une Sub à tout moment, il faut utiliser la commande Retrurn (déjà vue dans la routine Enregistrer sous).

Il est important de vérifier si le rôle du bouton cmdDefilement est assuré correctement. Pour s'en assurer, il suffit de mettre une alarme dans le Timer (la Sub Timer est vide pour l'instant), et de lancer le programme. D'abord, trouver un fichier son.wav quelque part, et ensuite, le faire lire dans la Sub du Timer. Le fichier son « Move.wav » est assez adéquat (bruit d'une prise de photo), mais n'importe quel autre à disposition fera l'affaire.

La mise en œuvre du code pour le son se présente comme ceci, dans la Sub du Timer :

```
PUBLIC SUB TimerDefile_Timer()
DIM AuSon AS Sound
AuSon = NEW Sound(Application.path & "/Datas/move.wav")
AuSon.Play
```

```
END
```

Naturellement, le fichier son (move.wav) a été copié dans le répertoire Datas, là où sont les paramètres.

Ce code écrit, il est utile de tester que la procédure fonctionne, en lançant le programme, puis un click sur la bouton « Départ », on doit entendre le son toutes les secondes. Le bouton étant positionné maintenant sur « Stop », un nouveau click doit arrêter le son.
Il est probable cependant qu'aucun son ne sorte des haut-parleurs ! En effet, pour que la fonction « Sound » fonctionne, il faut que le composant Gambas adéquat soit sélectionné. Si ça ne marche pas, un petit tour dans les Propriétés du Projet, troisième onglet, et cette fois, il faut sélectionner la ligne « gb.sdl.sound » et cocher sa case en début de ligne. Il est possible que cela soit un peu différent pour de nouvelles versions, ce sera à expérimenter.
Si tout fonctionne normalement, il faut ensuite aller lire la ListBox, et afficher la ligne concernée. Tout d'abord, il faut aller lire la ligne 1, sinon, il n'y aura rien d'affiché pendant le premier cycle du Timer. Ensuite, il faut arrêter le traitement à la dernière ligne, et pour s'y retrouver, il faudra un compteur. Donc, déclaration d'une nouvelle variable qui devra être générale à la feuille, donc, placée dans les déclarations du début : Private Conteur As Integer.
On aurait pu utiliser la fonction d'affichage de ligne déjà présente avec valBox. ValBox est une boîte de saisie chiffre, aussi, il suffit de récupérer son entrée pour afficher une ligne quelconque de la listBox. Il est aussi simple de rester dans la même géographie, du reste, il y a peu à rajouter.
Dans le premièrement du bouton, on va afficher la première ligne, comme dit plus haut, ensuite, tout se passera dans la Sub du Timer pour les lignes suivantes.
Dans le deuxièmement du bouton, il faut arrêter le Timer, c'est déjà fait, il suffit de remettre la variable Conteur à zéro, dans le cas d'une autre commande.
Dans le Timer, nous n'avons pour l'instant que le son. Il faut ajouter une incrémentation de 1 pour le « conteur », une ligne pour l'affichage (déjà vu), et ensuite, effectuer un test de reconnaissance de fin de liste. A ce propos, il suffira de se servir de la propriété « lstBoxNoms.Count - 1 », **moins un** car il ne faut pas oublier qu'on affiche avec l'index... Bien entendu, le processus « son » étant resté dans le Timer, un son sera émis à chaque nouvel affichage.

Un détail en passant : il semblerait que les variables locales (DIM) doivent impérativement être toutes déclarées dans le début de la Sub, sinon, on a un message d'erreur. Ceci semble une particularité de Gambas... Mais après tout, ça fait plus ordonné...
Il reste maintenant à utiliser la fonction de temps variable, accessible avec le second bouton radio. Il va donc falloir rajouter deux Sub, une pour chaque bouton, la première, fixe, déterminera la valeur Delay du Timer à 1000, la seconde accordera la valeur du Timer au curseur.
Pour le curseur, on va lui donner une borne basse à 500, une borne haute à 5 000, ce qui devrait donner un affichage entre une demie-seconde et cinq secondes. Il aura une Value initiale de 1000, un pas (Step) de 500. Il faut aussi penser à afficher sa valeur dans le lblTemps prévu à cet effet.
(Il semblerait que la durée du Timer diffère assez nettement du Delay affiché – à voir).
(Il semblerait aussi que le pas du Timer ne tienne aucun compte de la valeur renseignée dans sa propriété.)
Donc, une nouvelle Sub pour le Slider, avec la commande :

```
PUBLIC SUB sldTemps_Change()
  TimerDefile.Delay = sldTemps.Value
  IF sldTemps.Value < 2000 THEN lblTemps.Text = (sldTemps.Value / 1000) & " seconde"
```

```
 IF sldTemps.Value > 1999 THEN lblTemps.Text = (sldTemps.Value / 1000) & " secondes"
END
```

Ici, il est écrit deux lignes pour tenir compte du pluriel éventuel de « seconde »...
Et puis, pendant qu'on y est, on pourra aussi mémoriser la valeur du Slider, pour la retrouver dans la même position au lancement de l'application : rien de plus simple : il suffit d'ajouter « sldTemps.Value » dans les Sub lecture et écriture du fichier Parametres.

Attention placer la nouvelle instruction dans Lire, après avoir lancé l'application, sinon, c'est le message d'erreur. C'est toujours la même chose !

Bien entendu, il faut obligatoirement inhiber le bouton « cmdDefilement » si le nombre d'inscrits dans la liste est inférieur à 3, car la chose ne présenterait aucun intérêt. Le procédé est le même que pour le blocage des autres touches. Ceci doit pouvoir se mettre en place sans indications particulières !

A ce stade, votre code devrait ressembler à peu près à ce qui est présenté dans le chapitre suivant.

Chapitre 7

Voici, en son intégralité, le code définitif à la fin de la première partie de cette étude. Il n'a pas vocation à être recopié purement et simplement, il constitue en sorte un résumé permettant de trouver rapidement tel ou tel emploi d'un contrôle ou d'une commande.

```
' Gambas class file
PRIVATE Age AS Integer
PRIVATE Agee AS String
PRIVATE Nom AS String
PRIVATE Nome AS String
PRIVATE Erreur AS Boolean
PRIVATE Couleur AS String
PRIVATE Polisse AS String
PRIVATE FondForm AS String
PRIVATE SaveDonnees AS Boolean
PRIVATE Modif AS Boolean
PRIVATE Defil AS Boolean
PRIVATE Conteur AS Integer

PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()
  ME.Center
  Erreur = FALSE
  Modif = FALSE
  Defil = FALSE
```

```
  LireFichierParametres
  LireFichierDonnees
  ChargerData
  txtSaisieNom.Text = ""
  txtSaisieAge.Text = ""
  txtSaisieAge.Enabled = FALSE
  lblNombre.Text = lstBoxNoms.Count
  txtSaisieNom.SetFocus
END

PUBLIC SUB txtSaisieNom_Change()
  Nom = txtSaisieNom.Text
  IF Len(Nom) > 0 THEN
    Label1.Text = ""
    lblAffichage.Text = "Saisie en cours... "
    txtSaisieAge.Text = ""
  ENDIF
END

PUBLIC SUB txtSaisieNom_KeyPress()
  ' on récupère le code de la touche pressée "Key.Code"
  'que l'on compare à la constante "Key.RETURN"
  'Si il n'y a pas d'erreur, on affiche
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    IF txtSaisieNom.Text = "" THEN Erreur = TRUE
    IF txtSaisieNom.Text <> "" THEN VerifNom
    IF Erreur = FALSE THEN
      Affichage 'ex cmdAfficher_Click
      txtSaisieNom.Enabled = FALSE
      txtSaisieAge.Enabled = TRUE
      txtSaisieAge.SetFocus
    ENDIF
  ENDIF
  IF Erreur = TRUE THEN
    Label1.Text = "Veuillez entrer un nom correct, SVP..."
    txtSaisieNom.Text = ""
    Nom = ""
    Erreur = FALSE
    txtSaisieNom.SetFocus
  ENDIF
END

PRIVATE SUB VerifNom()
  'Vérifier que le nom ne porte que des lettres
  DIM N AS Integer
  FOR N = 1 TO Len(nom)
    IF Asc(Mid$(Nom, N)) > 45 AND Asc(Mid$(Nom, N)) < 64 THEN Erreur = TRUE
    IF Asc(Mid$(Nom, N)) > 32 AND Asc(Mid$(Nom, N)) < 45 THEN Erreur = TRUE
```

```
  NEXT
END

PUBLIC SUB txtSaisieAge_Change()
  txtSaisieNom.Enabled = FALSE
  'Vérifier que ce ne sont que des chiffres
  IF IsNumber(Val(txtSaisieAge.Text)) THEN Age = Val(txtSaisieAge.Text)
END

PUBLIC SUB txtSaisieAge_KeyPress()
    ' on récupère le code de la touche... Comme ci-dessus
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    IF Age > 29 AND Age < 56 THEN
      Affichage
      Label1.Text = ""
      ELSE
      Label1.Text = "Veuillez entrer un nombre correct, SVP..."
      txtSaisieAge.Text = ""
      txtSaisieAge.SetFocus
    ENDIF
  ENDIF
END

PRIVATE SUB Affichage() 'ex cmdAfficher_Click()
  IF Age < 30 THEN
    lblAffichage.Text = "Nom : " & Nom
    txtSaisieNom.Enabled = FALSE
    txtSaisieAge.Enabled = TRUE
  ENDIF
  IF Age > 29 THEN
    Agee = CStr(Age)
    lblAffichage.Text = "Nom : " & Nom & " - Age : " & Agee & " ans"
    lstBoxNoms.Add("Nom : " & Nom & " - Age : " & Agee & " ans")
    lblNombre.Text = lstBoxNoms.Count
    Modif = TRUE
    cmdEnregistrerListe.Enabled = TRUE
    IF lstBoxNoms.Count > 0 THEN
      cmdSupprimer.Visible = TRUE
      cmdVider.Visible = TRUE
      IF lstBoxNoms.Count > 2 THEN
        cmdDefilement.Enabled = TRUE
        lblNumligne.Visible = TRUE
        vlBox.Visible = TRUE
      ENDIF
    ENDIF
    IF Len(Nom) > 2 THEN Nome = Nom
    txtSaisieNom.Text = ""
    txtsaisieage.text = ""
```

```
    txtsaisieage.Enabled = FALSE
    Age = 0
    txtSaisieNom.Enabled = TRUE
    txtSaisieNom.SetFocus
  ENDIF
END

PUBLIC SUB Fond_Click()
  IF rdFond1.Value = TRUE THEN
    lblAffichage.Background = &HFFFF5F&
    Couleur = 1
  ENDIF
  IF rdFond2.Value = TRUE THEN
    lblAffichage.Background = &H00FFBF7F&
    Couleur = 2
  ENDIF
  IF rdFond3.Value = TRUE THEN
    lblAffichage.Background = &H009FFFCF&
    Couleur = 3
  ENDIF
END

PUBLIC SUB Police_Click()
  IF rdPolice1.Value = TRUE THEN
    lblAffichage.Font = Font["Normal"]
    Polisse = 1
  ENDIF
  IF rdPolice2.Value = TRUE THEN
    lblAffichage.Font = Font["Bold"]
    Polisse = 2
  ENDIF
  IF rdPolice3.Value = TRUE THEN
    lblAffichage.Font = Font["Italic"]
    Polisse = 3
  ENDIF
END

PUBLIC SUB cmdSupprimer_Click()
  'Supprimer l'élément sélectionné de la liste
  lstBoxNoms.Remove(lstBoxNoms.Index)
    IF lstBoxNoms.Count < 1 THEN
    cmdSupprimer.Visible = FALSE
    cmdVider.Visible = FALSE
    cmdEnregistrerListe.Enabled = FALSE
  ENDIF
  lblNombre.Text = lstBoxNoms.Count
  Modif = TRUE
  IF lstBoxNoms.Count < 3 THEN
```

```
    cmdDefilement.Enabled = FALSE
    lblNumligne.Visible = FALSE
    vlBox.Visible = FALSE
  ENDIF
END

PUBLIC SUB cmdVider_Click()
  lstBoxNoms.clear
  cmdSupprimer.Visible = FALSE
  cmdVider.Visible = FALSE
  lblNombre.Text = "0"
  Modif = TRUE
  cmdEnregistrerListe.Enabled = FALSE
  cmdDefilement.Enabled = FALSE
  lblNumligne.Visible = FALSE
  vlBox.Visible = FALSE
END

PUBLIC SUB lstBoxNoms_Click()
  lblAffichage.Text = lstBoxNoms.Text
  cmdSupprimer.Visible = TRUE
  cmdVider.Visible = TRUE
END

PUBLIC SUB vlBox_KeyPress()
  DIM Nbre AS Integer
  Nbre = vlBox.Text - 1
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    'lblAffichage.Text = lstBoxNoms.List[Nbre] ' 1ère méthose : directe
    lstBoxNoms.Index = Nbre        ' 2 ème méthode : va activer lstBoxNoms_Click
  ENDIF
END

PUBLIC SUB CreerFichierParametres()
  DIM FichierData AS File
  DIM Save AS Byte
  IF SaveDonnees = TRUE THEN Save = 1
  IF SaveDonnees = FALSE THEN Save = 0
  FichierData = OPEN Application.path & "/Datas/Parametres.txt" FOR CREATE
  PRINT #FichierData, Couleur
  PRINT #FichierData, Polisse
  PRINT #FichierData, Save
  PRINT #FichierData, sldTemps.Value
  CLOSE #FichierData
 END

PUBLIC SUB LireFichierParametres()
  DIM FichierData AS File
```

```
  DIM Save AS Byte
  FichierData = OPEN Application.path & "/Datas/Parametres.txt" FOR READ
  LINE INPUT #FichierData, Couleur
  LINE INPUT #FichierData, Polisse
  LINE INPUT #FichierData, Save
  LINE INPUT #FichierData, sldTemps.Value
  CLOSE #FichierData
  IF Save = 1 THEN SaveDonnees = TRUE
  IF Save = 0 THEN SaveDonnees = FALSE
CATCH
  Message.Info(Error.Text)
 END

PUBLIC SUB cmdEnregistrerListe_Click()
  'Creer la chaîne de Donnees
  DIM N AS Integer
  DIM Aenregistrer AS String
  Aenregistrer = lstBoxNoms.List[0]
  FOR N = 1 TO lstBoxNoms.Count - 1
  Aenregistrer = Aenregistrer & Chr$(13) & lstBoxNoms.List[N]
  NEXT
  'Appeler la boîte de dialiogue
  Dialog.Filter = ["*.txt", "Text Files"]
  IF Dialog.SaveFile() THEN RETURN
  File.Save(Dialog.Path, Aenregistrer)
CATCH
  Message.Info(Error.Text)
END

PUBLIC SUB CreerFichierDonnees()
  DIM FichierData AS File
  DIM N AS Integer
  IF lstBoxNoms.Count < 1 THEN RETURN
  FichierData = OPEN Application.path & "/Datas/Donnees.txt" FOR CREATE
  FOR N = 0 TO lstBoxNoms.Count - 1
  PRINT #FichierData, lstBoxNoms.List[N]
  NEXT
  CLOSE #FichierData
 END

 PUBLIC SUB LireFichierDonnees()
  DIM FichierData AS File
  DIM Ligne AS String
  FichierData = OPEN Application.path & "/Datas/Donnees.txt" FOR READ
  WHILE NOT Eof(FichierData)
  LINE INPUT #FichierData, Ligne
  lstBoxNoms.Add(Ligne)
```

```
  WEND
  CLOSE #FichierData
  cmdEnregistrerListe.Enabled = TRUE
 CATCH
  Message.Info(Error.Text)
 END

PUBLIC SUB ChargerData()
  SELECT CASE Couleur
    CASE 1
    rdFond1.Value = TRUE
    CASE 2
    rdFond2.Value = TRUE
    CASE 3
    rdFond3.Value = TRUE
  END SELECT
  SELECT CASE Polisse
      CASE 1
    rdPolice1.Value = TRUE
      CASE 2
    rdPolice2.Value = TRUE
      CASE 3
    rdPolice3.Value = TRUE
  END SELECT
  IF SaveDonnees = TRUE THEN chkSave.Value = TRUE
  IF SaveDonnees = FALSE THEN chkSave.Value = FALSE
  IF lstBoxNoms.Count > 2 THEN
    cmdDefilement.Enabled = TRUE
    lblNumligne.Visible = TRUE
    vlBox.Visible = TRUE
  ENDIF
END

PUBLIC SUB chkSave_Click()
  IF chkSave.Value = TRUE THEN SaveDonnees = TRUE
  IF chkSave.Value = FALSE THEN SaveDonnees = FALSE
END

PUBLIC SUB cmdDefilement_Click()
  IF Defil = FALSE THEN 'Pour afficher
    Defil = TRUE
    cmdDefilement.Text = "Stop"
    TimerDefile.Enabled = TRUE
    'Afficher le premier élément
    lblAffichage.Text = lstBoxNoms.List[0]
    RETURN
  ENDIF
  IF Defil = TRUE THEN 'Pour arrêter
```

```
    Defil = FALSE
    cmdDefilement.Text = "Départ"
    TimerDefile.Enabled = FALSE
    Conteur = 0
  ENDIF
END

PUBLIC SUB TimerDefile_Timer()
  DIM AuSon AS Sound
  Conteur = Conteur + 1
  lblAffichage.Text = lstBoxNoms.List[Conteur]
  AuSon = NEW Sound(Application.path & "/Datas/move.wav")
  AuSon.Play
  IF Conteur = lstBoxNoms.Count - 1 THEN
    Conteur = 0
    TimerDefile.Enabled = FALSE
    cmdDefilement_Click
  ENDIF
END

PUBLIC SUB rdFixe_Click()
  TimerDefile.Delay = 1500
END

PUBLIC SUB rdVariable_Click()
  TimerDefile.Delay = sldTemps.Value
END

PUBLIC SUB sldTemps_Change()
  TimerDefile.Delay = sldTemps.Value
  IF sldTemps.Value < 2000 THEN lblTemps.Text = " Temps : " & (sldTemps.Value / 1000) & "
seconde"    ' Tout sur la même ligne !
  IF sldTemps.Value > 1999 THEN lblTemps.Text = " Temps : " & (sldTemps.Value / 1000) & "
secondes"    ' Tout sur la même ligne !
END

PUBLIC SUB cmdQuitter_Click()
   DIM Reponse AS Integer
   CreerFichierParametres
   IF SaveDonnees = TRUE THEN
    CreerFichierDonnees
    Message.Info("Données enregistrées")
    ELSE
    IF Modif = TRUE THEN
       Reponse = Message.Warning("Vos données ont été modifiées, voulez-vous les enregistrer ?
", "Enregistrer", "Ignorer")    ' Tout sur la même ligne !
       IF Reponse = 1 THEN
         CreerFichierDonnees
```

```
        Message.Info("Données enregistrées")
        ELSE
        Message.Info("Données non enregistrées")
       ENDIF
    ENDIF
  ENDIF
  ME.Close
  QUIT
END
```

Voilà présentés plusieurs contrôles et leur mise en œuvre. Cela permet déjà d'écrire quelques petites applications.
Cela dit, il faut sauvegarder ce travail, c'est le moment de créer un sous-répertoire « Apprentissage002 », et d'y coller tout ce qui est en « Apprentissage001 » !
Tout le code qui a été créé et modifié est intégralement recopié ci-dessus, et cela prend déjà 6 pages.
Désormais et à l'avenir, seules les nouvelles lignes seront présentées.
A titre indicatif et tout à fait exceptionnel, voici une copie d'écran de l'application qui tourne. A noter, une mauvaise entrée Age, et le message d'avertissement.

Comme déjà dit, le code ci-dessus est largement perfectible. Cela dit, son seul intérêt, c'est de permettre de vérifier un point plus ou moins bien expliqué dans le texte. Si des améliorations doivent être apportées, c'est à chacun de le faire, dans son propre code !

--- Fin de la première partie ---

Seconde partie

Chapitre 1

Pour commencer cette seconde partie, j'ai décidé de passer à la version supérieure de Gambas, soit la 2.19. Ce qui suit concerne une distribution Ubutu, je ne sais pas pour les autres (mais ça devrait être assez ressemblant).

D'abord, trouver le fichier adéquat soit le compressé :

http://prdownloads.sourceforge.net/gambas/gambas2-2.19.0.tar.bz2?download

Une fois le « tar.bz » récupéré, sur le bureau ou ailleurs, il suffit de le décompresser dans home, ce qui va créer un répertoire du même nom (Gambas2-2.19.0). Cette étape effectuée, il faut désinstaller l'actuel Gambas à l'aide du gestionnaire Synaptic, et opter pour une désinstallation totale.

L'étape suivante se passe dans le terminal.

Aller dans le répertoire par « cd Gambas-2.19. »

Ensuite taper les instruction suivantes :

- ./configure (c'est une opération assez longue...)
- make
- sudo make install

Normalement, quand le terminal rend la main, Gambas devrait être installé, il suffit de taper « Gambas2 » dans la console. Si tout s'est bien passé, Gambas devrait être opérationnel. Cependant, il n'apparaît pas dans les menus, on peut, soit l'y installer, soit créer un lanceur, c'est ce que j'ai fait, le lanceur me suffit. Pour le lanceur, il faut récupérer l'icône (new-logo.png) dans le répertoire créé, et la copier dans le répertoire spécifique : /usr/share/pixmaps/new-logo.png

Si la copie n'est pas possible (droits non alloués), passer par la console en utilisant :

sudo cd /lieude/new-logo.png /usr/share/pixmaps/new-logo.png

Bien, on suppose que Gambas 2.19 est installé et opérationnel. Normalement, et si le répertoire de travail Gambasse a été installé en home, tous les anciens fichiers doivent être présents au lancement, en particulier, le tout neuf « Apprentissage002 ». C'est donc ici qu'il faut reprendre.

Jusqu'à présent, tout s'est passé sur une seule feuille, la feuille Saisies, avec, d'une part, la feuille graphique, Form, et la feuille de code, Class. Cela peut bien entendu être suffisant pour nombre d'applications.

On peut cependant aimer avoir une feuille qui apparaît au lancement, et qui permet d'en appeler d'autres. On peut simplement dire que c'est une question de préférences. Nous allons voir comment rajouter une feuille.

Chapitre 2

Pour ajouter une feuille – un formulaire, rien de plus simple : un clic droit dans l'arborescence de gauche, un menu contextuel propose « Nouveau ». Choisir « Formulaire », et une boîte de dialogue demande le nouveau nom, ici ce sera « Accueil ». Ensuite, clic droit sur Accueil dans l'arborescence de gauche, et dans le menu, cliquer sur « Classe de démarrage ». Désormais, au lancement de l'application, c'est la page Accueil qui va être lancé à l'ouverture. Quand on lance le programme, on voit que c'est bien cette feuille qui est lancée.

Ensuite, il paraît utile de renommer la feuille d'origine, FMain, d'autant qu'elle n'est plus prioritaire au lancement. Avec la version 2.19, un simple clic droit, et Renommer, fera l'affaire. Avec la 2.13, la chose ne semble pas si bien fonctionner. Pour ma part, j'ai contourné le problème en arrêtant tout, et en changeant les noms en utilisant le gestionnaire de fichiers, Nautilus sous Ubuntu. Mais il est certain que le faire directement est plus pratique : une autre bonne raison de passer à la 2.19.

Donc, j'ai renommé cette feuille (ou formulaire) du nom de « Saisies » puisqu'en fait, il y avait essentiellement de la saisie.

La première chose à mettre sur cette nouvelle feuille Accueil, c'est naturellement un bouton de sortie. Cela dit, il faut considérer cette feuille comme un accueil, la bienvenue, et donc, il faut un minimum de présentation. Je place donc, sur la feuille, un« Panel », et que, comme tout contrôle, je vais chercher dans la boîte à outils (onglet Container).

(Au passage, si boîte à outils est fenêtre droit ont disparu de l'écran, un clic sur le petit triangle, juste dans l'horizontale des onglets les remettra à place !)

Donc, je mets mon panel sur la feuille, je laisse le nom Panel1, et sur ce panel, je pose un label pour écrire « Bienvenue au club », ce qui n'a rien d'original. Une petite mise en page pour cadrer tout ça, agrandir la police, etc, au besoin, un peu de couleur de fond avec Background. Ensuite, je dis que cette feuille au fond, pourrait servir de « fond d'écran » pour toute l'application, aussi, je mets la propriété Maximized de la feuille à True. Attention et pour l'instant, je ne mets surtout pas à True la propriété FullScreen, j'aurais grands risques de soucis. Il est même important de vérifier que FullScreen est bien à False, avant tout lancement.

Si on lance l'application, la feuille Accueil s'affiche normalement, avec son bandeau est ses 3 cases système, ce qui permet d'arrêter le logiciel. Mais on remarque que le panel et sa Bienvenue, se balade un peu n'importe où. Le mieux serait de le centrer, et ce, qu'elle que soit la taille de la fenêtre. Il y a une formule magique pour tout centrer, et elle est très simple. Il faut toujours garder à l'esprit que les contrôles se définissent géographiquement par leur Hauteur (Height), leur Largeur (Width), et le point d'origine, soit le coin haut et gauche du contrôle, point précisé chez Gambas par X et Y (Left et Top en VB). Donc, et si on veut placer un contrôle au milieu d'une feuille, ou d'un Panel comme ici, il suffit se soustraire les dimensions et de diviser par deux, pour avoir les origines du point. Il faut aussi bien garder à l'esprit que c'est le support qui sert de référence, et le support, ce n'est pas forcément la feuille !

Maintenant, que faire si la taille du conteneur vient à changer ? Facile, on utilise sa propriété « Resize ». Ainsi, et dans le cas présent, pour maintenir mon panel au milieu de ma feuille Accueil, je vais coder ceci :

```
PUBLIC SUB Form_Resize()
    Panel1.X = (Accueil.Width - Panel1.Width) / 2
```

```
    Panel1.Y = (Accueil.Height - Panel1.Height) / 2
END
```

Si pour une raison ou une autre, la taille change, les valeurs de X et Y changeront en conséquence.

Cela dit, si ça fonctionne bien, il y a quand même un petit défaut d'aspect, le panel a tendance à se balader un peu au lancement, avant de se centrer comme il faut. Si on veut jouer les perfectionnistes, on peut tourner simplement la difficulté : on met la propriété visible du panel à False, et dans la commande Open de la feuille, on lance un Timer, lequel timer sera réglé à 100 par exemple, soit 1/10ème de seconde. Dans la Sub du Timer, d'abord, on l'arrête, et ensuite, on met à True la propriété visible du panel, lequel s'est mis en place pendant ce temps-là... Bon, c'est juste une amusette...

Une fois cette présentation modeste mise en place, il faut mettre deux choses importantes sur cette feuille : un bouton Quitter, et un bouton pour accéder aux autres feuilles, celle existante, Saisies, et celle(s) à venir.

Pour le bouton Quitter, il serait bon qu'une procédure soit accessible pour tout le monde. Cela est possible, il suffit de la mettre, non pas dans un formulaire, mais dans un Module et de pouvoir l'appeler depuis n'importe quelle feuille. Rendez-vous donc, dans l'arborescence de gauche, clic droit, Nouveau / Module, lequel va être appelé « General ». (Au passage : jamais de lettres accentuées dans le code). Dans ce module, on va écrire une Sub qui va fermer tous les formulaires et terminer par la commande Quit, histoire de terminer propre. Ainsi, il suffira d'appeler cette Sub depuis n'importe quelle feuille pour arrêter proprement le programme. Pour l'instant, le module General devrait donc ressembler à ceci :

```
' Gambas module file
PUBLIC SUB Fermer()
  Accueil.Close
  Saisies.Close
  QUIT
  END
```

Ainsi, et dans chaque feuille, il suffira d'appeler cette Sub par la commande :

```
General.Fermer
```

En effet, quand on appelle quelque chose d'un autre endroit, il faut préciser d'abord cet endroit, puis le compléter pas la commande adéquate, ici le nom de la procédure.

Maintenant, il faut pouvoir appeler la feuille (et les futures feuilles), depuis la page Accueil. Un bouton, à mettre sur le feuille Accueil et avec un double Clic déjà comme déjà vu, il suffira de mettre le nom de la feuille à appeler, suivie de la commande Show, ce qui donnera par exemple :

```
PUBLIC SUB cmdSaisies_Click()
  Saisies.Show
```

```
END
```

Si on est soucieux d'une certaine logique, il serait normal de faire la même chose pour retourner à la feuille Accueil quand on est dans la feuille Saisies. Là encore, même méthode, un bouton à rajouter. En résumé chaque feuille devrait contenir autant de boutons d'appel qu'il y a de feuilles, avec, en plus, un bouton d'arrêt. Pas de consignes particulières, c'est toujours la même chose.

Pour continuer, on va rajouter une troisième feuille : un clic sur l'arborescence à gauche « Nouveau / Formulaire » et on a une belle feuille toute neuve. On va lui donner le nom de « Sites », et y ajouter 3 boutons, respectivement Quitter, Saisies, Accueil, de manière à pouvoir retourner dans n'importe quelle feuille, comme on l'a fait pour les deux précédentes. Le bouton Quitter va appeler la fermeture dans le module General, comme ci-dessus. Normalement et à ce stade, on doit pouvoir naviguer d'une feuille à l'autre, et pouvoir sortir depuis n'importe quelle feuille. Pour faire « joli », on peut aussi donner une couleur de fond à la feuille, on peut aussi donner à nos trois feuilles les mêmes cotes : hauteur, largeur, et centré. Ainsi, on aura une apparence d'uniformité en passant de l'une à l'autre.

Avant d'aller plus loin, il faut faire un petit exercice de réflexion : si on ajoute un nouveau nom depuis la feuille Saisies, puis si ensuite, on retourne sur Accueil, et que l'on quitte le programme, notre dernière saisie est tout simplement perdue ! Il va donc falloir ruser...

La feuille site avait son code établi pour enregistrer les fichiers à la fermeture du programme. Mais si on quitte le programme ailleurs, il faut revenir vérifier que l'enregistrement a bien eu lieu, le faire s'il n'a pas été fait.

Pour ce faire, on va tout simplement modifier la PUBLIC SUB cmdQuitter_Click(), et la renommer en Quitter. Ensuite, et par un double-clic sur la bouton Quitter, on va refaire une nouvelle PUBLIC SUB cmdQuitter_Click(), laquelle va appeler la Sub Fermer, dans le module , comme indiqué ci-dessus. On voit donc que, à la fermeture depuis le feuille Saisies, le logiciel va aller dans le module pour exécuter la procédure Fermer. Et c'est justement dans cette procédure Fermer qu'on va rajouter, en première ligne, un appel à la procédure Quitter, laquelle va faire le nécessaire, c'est-à-dire, enregistrer (systématiquement) le fichier Paramètres, et éventuellement le fichier Données.txt, si des modifications ont été apportées. Ensuite, le logiciel va retourner dans le Module pour terminer la procédure « Fermer ». Ainsi, la sauvegarde sera obligatoirement faite, quelque soit la feuille d'où l'on appellera la commande de fermeture. Si par la suite, on a d'autre(s) sauvegarde(s) à faire, il faudra la(es) ajouter en début de la Sub, ou immédiatement après Saisies.Quitter.

Dans le Module, la Sub, la procédure Fermer devrait ressembler à ceci :

```
PUBLIC SUB Fermer()
  Saisies.Quitter ' Va exécuter la procédure dans la feuille Saisies
  Accueil.Close ' Ferme cette feuille
  Saisies.Close ' Ferme cette feuille
  Sites.Close ' Ferme cette feuille
  QUIT ' Ferme l'application
END
```

Chapitre 3

Cette feuille, appelée « Sites », va contenir la liste des pays susceptibles d'avoir besoin d'un capitaine, par exemple, l'un de ceux inscrits dans le fichier, fichier constitué dans la feuille Saisies. Comme il s'agit de confier le commandement à une personne pouvant être de nationalité de l'un des pays adhérents, on considère que chaque pays a son règlement, quand à l'âge du capitaine et les limites sont différentes d'un pays à l'autre, certains gouvernements verront les limites d'âge à des niveaux différents.

On va donc porter sur la nouvelle feuille « Sites » deux listBox, pour rentrer, dans l'ordre, l'âge maximum et ensuite, l'âge minimum requis. Pour l'instant, on va se contenter de 6 pays, le mieux pour les reconnaître, c'est encore d'afficher leur pavillon, et pour les sélectionner, on prendra 6 boutons radio, bien placés sur une frame dédiée. Un label va permettre d'écrire en clair le nom du pays sélectionné, de manière à ne pas faire d'erreur !

Pour afficher les drapeaux des pays, il faut d'abord les avoir ! J'ai trouvé les images adéquates sur le lien suivant :

http://cliparts.toutimages.com/drapeaux/europe/page_1.htm

et j'ai pris : Allemagne, Belgique, Espagne, France, Italie, Portugal. Ils sont affichés dans une PictureBox, un composant pas encore utilisé. Il est bien entendu dans la boîte à outils, se présente comme un crevette et une fois sur la feuille, on le met à la bonne dimension. Je l'ai nommé « picDrapeau » et il y a une chose à ne pas oublier, c'est de mettre sa propriété Strech à True. En effet, cette propriété à pour grand intérêt de faire coller l'image contenue au contenant, et si elle est à False, l'image gardera sa taille, et qui ne sera pas forcément la même que celle de la pictureBox. Donc, aucune difficulté pour ce nouveau contrôle, pas davantage pour les boutons radio qui vont commander l'affichage. Bien entendu, il faudra « loger » les 6 images.png dans un dossier adéquat, comme l'application comporte un dossier Datas et un dossier Images, j'ai rajouté un sous-dossier au sous-dossier Images, et je l'ai appelé « Drapeaux ». Pour appeler une image, dans une pictureBox, il suffit de mettre la commande suivante :

```
picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/allemagne.gif"] .
```

Attention, en Gambas, les arguments sont entre [ ], et non ( ). Les images sont appelées depuis les Sub des boutons radio, lesquels donnent aussi à l'occasion le nom du pays dans le label correspondant. A noter que les boutons radio ne sont pas indexables comme en VB, il faut donc une Sub séparée pour chaque bouton (on y reviendra plus loin).

Pour les textBox, le principe est le même que dans la feuille Saisie, quant à l'âge, il va juste y avoir quelques petits arrangements à faire. J'ai nommé la première txtAgeMax, l'autre, txtAgeMin. Deux labels par devant pour préciser ce et quoi, un label en dessous pour info : d'abord, afficher « Et validez par Return » (même procédé qu'en Saisises), et ensuite, lors de la saisie, signaler si il y a erreur. Bien entendu, si erreur il y a, il faut vider la case, et lui redonner le focus, si la première saisie est correcte, il faut donner le focus à la seconde, soit « Entrez l'âge minimum ». Un petit détail à noter : pour la saisie de l'âge minimum, il ne peut être que d'au moins un an de l'âge maximum ! Comme pour la feuille Saisies, j'ai utilisé deux Sub, une Sub_Change pour obtenir la valeur, une Sub_KeyPress pour intercepter le Return.

Il faut maintenant vérifier que les entrées fonctionnent normalement, et débogguer s'il y a problème.

A ce stade, votre code devrait ressembler à ceci :

```
' Gambas class file
PUBLIC Pays AS String
PUBLIC AgeMax AS Integer
PUBLIC AgeMin AS Integer

PUBLIC SUB Form_Open()
  ME.Center
  rdPays4_Click ' Pour mettre la France par défaut
  txtAgeMax.SetFocus
END

PUBLIC SUB cmdSaisies_Click()
  Saisies.Show
END

PUBLIC SUB cmdAccueil_Click()
  Accueil.Show
END

PUBLIC SUB Quitter_Click()
   General.Fermer
END

PUBLIC SUB rdPays1_Click()
  Pays = "Allemagne"
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/allemagne.gif"] '
  lblAffichPays.Text = Pays
END
PUBLIC SUB rdPays2_Click()
  Pays = "Belgique"
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/belgique.gif"] '
  lblAffichPays.Text = Pays
END
```

```
PUBLIC SUB rdPays3_Click()
  Pays = "Espagne"
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/espagne.gif"] '
  lblAffichPays.Text = Pays
END
PUBLIC SUB rdPays4_Click()
  Pays = "France"
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/france.gif"] '
  lblAffichPays.Text = Pays
END
PUBLIC SUB rdPays5_Click()
  Pays = "Italie"
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/italie.gif"] '
  lblAffichPays.Text = Pays
END
PUBLIC SUB rdPays6_Click()
  Pays = "Portugal"
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/portugal.gif"] '
  lblAffichPays.Text = Pays
END

PUBLIC SUB txtAgeMax_Change()
  lblInforme.Text = "Et validez par Return"
  'Vérifier que ce ne sont que des chiffres
  IF IsNumber(Val(txtAgeMax.Text)) THEN
    AgeMax = Val(txtAgeMax.Text)
    ELSE
    txtAgeMax.Text = ""
    lblInforme.Text = "Nombre incorrect"
  ENDIF
END

PUBLIC SUB txtAgeMin_Change()
  lblInforme.Text = "Et validez par Return"
```

```
 'Vérifier que ce ne sont que des chiffres
 IF IsNumber(Val(txtAgeMin.Text)) THEN
   AgeMin = Val(txtAgeMin.Text)
   ELSE
   txtAgeMin.Text = ""
   lblInforme.Text = "Nombre incorrect"
 ENDIF
 END

PUBLIC SUB txtAgeMax_KeyPress()
 ' on récupère le code de la touche... Comme ci-dessus
 IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
   IF AgeMax > 30 AND AgeMax < 56 THEN
     lblInforme.Text = "Entrée correcte"
     txtAgeMin.SetFocus
     ELSE
     lblInforme.Text = "Entrée incorrecte"
     txtAgeMax.Text = ""
     txtAgeMax.SetFocus
   ENDIF
 ENDIF
END

PUBLIC SUB txtAgeMin_KeyPress()
 ' on récupère le code de la touche... Comme ci-dessus
 IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
   IF AgeMin > 29 AND AgeMin < (AgeMax - 1) THEN
     lblInforme.Text = "Entrée correcte"
     ELSE
     lblInforme.Text = "Entrée incorrecte"
     txtAgeMin.Text = ""
     txtAgeMin.SetFocus
   ENDIF
 ENDIF
```

```
END
```

Bien. Cela posé, il serait bon de conserver ces données dans le but de les utiliser. D'abord, on va les mettre en mémoire à l'aide d'un tableau, puis les afficher avec une « GridView », et aussi les enregistrer.

D'abord, déclarer un tableau à deux dimensions, en début de feuille :

```
PUBLIC MaxiMini AS Integer[6, 2]
```

le 6 correspondant aux 6 pays, le 2 correspondant respectivement à l'âge Maxi et l'âge Mini.

Là encore, les références au tableau partent de zéro, donc, on « localise » de 0 à 5, et de 0 à 1. Il est déclaré comme Integer, puisque recevant des nombres.

J'ai eu quelques soucis avec les tableaux et je ne m'explique pas trop pourquoi. J'ai fait plusieurs essais dans l'application même, et j'avais des erreurs à répétitions. La dernière en date, et qui m'a bloqué un certain temps, venait du nom même de mon tableau : En effet, j'ai utilisé en toute simplicité MaxMin (puisqu'il s'agit en l'occurrence de maximum et de minimum), et je suppose que Gambas a pris cela pour des mots réservés. Il faut donc prendre bien garde quand on baptise nos variables, il y a des fois des surprises.... Cela étant, avec le nom ci-dessus, ça fonctionne.

Donc, et quand on va saisir une donnée – avec les deux txtBox prévues, on va faire deux choses en plus : la première, enregistrer la valeur (correcte) dans le tableau, la seconde, l'afficher dans une GridView.

Voyons pour celle-ci, car c'est un composant nouveau : un petit tour dans la boîte à outils, et on repère très vite notre composant, on l'amène sur la feuille, on le redimensionne un peu, on peut aussi le renommer (ce que je n'ai pas fait ici) et mettre sa propriété « Header » à « Horizontal », ce qui aura pour effet de permettre de mettre un titre aux colonnes (Pour titrer les lignes, il faut mettre à Vertical). Pour tout le reste ou presque, tout se commande par le code. Et comme ça ne s'invente pas, voici le code :

```
'Construire le CridView
 GRIDVIEW1.COLUMNS.Count = 2
 GRIDVIEW1.Rows.Count = 7
 GRIDVIEW1.COLUMNS[0].Width = 40
 GRIDVIEW1.COLUMNS[1].Width = 40
 GRIDVIEW1.Rows[0].Height = 30
 GRIDVIEW1.Rows[1].Height = 28
 GRIDVIEW1.Rows[2].Height = 28
 GRIDVIEW1.Rows[3].Height = 28
 GRIDVIEW1.Rows[4].Height = 28
 GRIDVIEW1.Rows[5].Height = 28
 GRIDVIEW1.COLUMNS[0].text = "Max"
```

```
GRIDVIEW1.COLUMNS[1].text = "Min"
```

Les valeurs de hauteur et de largeur ne sont pas impératives, elles correspondent à une présentation.

Ce code est à placer dans l'Open_Form de départ, de manière à être pris en considération à l'ouverture de la feuille.

Dans ce cas précis, j'ai placé la gridview juste à côté de la frame qui contient les 6 boutons des 6 villes. Ainsi, et si on s'arrange correctement, on va trouver les lignes de la gridview juste en face des boutons radio, il tombera donc sous le sens que chaque valeur affichée correspondra à la ville située sur le même niveau horizontal. Par ailleurs, je nomme les deux colonnes « Maxi et Mini », ce qui fait que l'on voit d'emblée à quoi les chiffres correspondent, sans autre affichage superflu. Il ne faut pas oublier que chaque ville est validée par le bouton radio correspondant, et comme l'ordre est établi de 0 à 5, il faut considérer que toute entrée d'âge maxi ou mini correspond bien avec la ville sélectionnée. En clair, le niveau « 3 » par exemple, correspondra à la ville 3 (en quatrième position !), mais aussi à AgeMax3 et AgeMin3 et tableau MaxMin[3,x], ici, x valant 0 ou 1. Cela permet de rentrer les valeurs dans le désordre, on n'est pas obligé de commencer par le premier pays, ici, l'Allemagne. Et donc, le code ci-dessus concernant les boutons radio n'est plus valable, le premier n'est plus « 1 » mais zéro. Cela n'était pas indispensable mais simplement logique, et par ailleurs, en plus de la variable « Pays », j'ai ajouté la variable « NumPays » (qui ira de 0 à 5), ce qui devient indispensable pour manipuler, et le tableau, et la gridview (et l'affichage que nous verrons plus loin). Par ailleurs, et à chaque appel de bouton radio, il y a des commandes obligatoires, telles l'effacement des txtBox de saisie, remettre le focus sur la txtbox AgeMax, afficher la ville sélectionnée (et d'autres choses à venir). J'ai réuni tout cela dans une nouvelle Sub qui est appelée à chaque bouton radio.

```
PRIVATE SUB AfficheRadio()
  lblAffichPays.Text = Pays
  txtAgeMax.Text = ""
  txtAgeMin.Text = ""
  lblInforme.Text = "Et validez par Return"
  txtAgeMax.SetFocus
END
```

Il reste naturellement à renseigner le tableau et la gridview à la fin de chaque saisie, soit après l'entrée correcte de AgeMax et ensuite de AgeMin. Cela va donc se faire tout naturellement dans les deux procédures, AgeMax_KeyPress, et AgeMin_KeyPress.

Enfin, il faut penser à enregistrer le tableau, pas la peine de re-rentrer les données à chaque fois.

Un bouton « cmdEnregistrer » va faire l'affaire, sachant que pour enregistrer le tableau, il faut prévoir deux boucles puisque c'est un tableau à deux dimensions !

```
PUBLIC SUB cmdEnregistrer_Click()
 'Enregistrer le tableau
 DIM FichierData AS File
  DIM N AS Integer
```

```
  DIM N1 AS Integer
  FichierData = OPEN Application.path & "/Datas/AgesMaxiMini.txt" FOR CREATE
  FOR N = 0 TO 5
    FOR N1 = 0 TO 1
      PRINT #FichierData, MaxiMini[N, N1]
    NEXT
  NEXT
  CLOSE #FichierData
CATCH
  Message.Info(Error.Text)
END
```

Ici, il est prévu une sécurité (Catch), c'est toujours utile de prévoir ce genre de chose dans les fichiers, ça évite un plantage en cas de fichier défectueux.

Enfin, un second bouton de lecture – cmdLecture_Click(), va tout simplement faire l'opération inverse. On pourrait prévoir le chargement automatique de ce fichier à l'ouverture de la feuille, il suffirait de mettre « cmdLecture_Click() » dans la première Sub de la feuille (PUBLIC SUB Form_Open()), mais cela n'est pas vraiment nécessaire ici. Cela dit, ce n'est pas interdit !

L'étape suivant va consister à trouver les candidats potentiels.

Il va donc falloir « scanner » le fichier des « Donnees.txt », récupérer l'âge de chaque inscrit, et le comparer aux Max et Min exigés par le pays concerné. Et cela pour chaque ligne du fichier de données. Seulement voilà, le fichier en question a enregistré une succession de phrases complètes, du genre « Nom : William - Age : 33 ans ». Et là, ce qu'il nous faut, c'est le « 33 », pas autre chose.

Ceci est encore l'occasion de dire qu'il faut bien réfléchir avant de se lancer dans le code, réfléchir à ses données, et à leurs noms, cf. mon souci avec le tableau ci-dessus et essayer de prévoir les aléas que cela pourrait entraîner : pas toujours facile...

Comme il n'est pas question de tout refaire, il va falloir ruser, cela va donner l'occasion de se servir des commandes liées au type String. Ici, on peut remarquer que toutes les lignes se terminent par « xx ans », ce qui signifie que la partie qui nous intéresse débute systématiquement à 6 caractères à partir de la droite : ou peut donc la récupérer facilement à l'aide de Right$(« phrase entière », 6). Une fois cela obtenu, il est facile de voir que les chiffres sont les deux premiers à gauche : ou peut donc les récupérer facilement à l'aide de Left$(« xx ans », 2. Enfin, avec un Val(xx), on récupère notre nombre qu'il sera facile de comparer avec le AgeMax et AgeMin de la ville sélectionnée par le bouton radio. Au fond, ce n'est pas si compliqué que ça...

Les données sont normalement stockées dans la lstNoms, dans la feuille Saisies, elles ne sont donc pas directement accessibles dans la feuille Sites. Une solution, parmi d'autres, va consister à lire directement le fichier et à extraire les informations ligne après ligne. Bien entendu, on pourra faire la comparaison directement après lecture, et si la personne répond aux critères, on l'affiche dans une listBox, sinon, on l'oublie, et on passe à la suivante.

Par souci de clarté, je vais faire une nouvelle procédure de sélection, et qui sera appelée directement après la lecture d'une ligne. Pour « lancer » la sélection, il faudra bien entendu mettre un nouveau bouton sur la feuille, son clic démarrera le processus. Ensuite, et pour la listBox de réception, il faudra prévoir son effacement par un bouton, mais aussi automatiquement dès sélection d'une autre ville. Enfin, et toujours pas souci de clarté, il faudra mettre un label au-dessus le la listBox de manière à rappeler que cette sélection est commandée pour la ville sélectionnée.

Bien, on va essayer de mettre tout ça en place !

D'abord, on met en place une listBox sur la feuille : on peut l'appeler « lstBoxNoms », tout comme celle utilisée dans la feuille des Saisies : pas d'interférences à craindre, chacun reste dans son monde. Au-dessus, j'installe un label, que j'appelle « lblSelect », et qui servira à rappeler le pays concerné, et donc, objet de la recherche. En dessous de la listBox, je place un autre label « lblCommentaire », lequel servira à afficher le nombre de résultats trouvés. Enfin, il faut mettre un bouton « cmdSelection », destiné à lancer la recherche, et un autre bouton « cmdVider », pour vider la listBox une fois la recherche terminée et visualisée. Les mises en place et ornements pour ces accessoires sont purement personnels.

Dans la cmdSelection », on va d'abord commencer à renseigner la labelBox du dessus pour afficher la ville. Ensuite, et si par hasard, le fichier des âges Maxi et Mini n'était pas lu, la recherche serait impossible, donc, par la peine d'aller plus loin : on met un message d'avertissement, et on sort de la Sub (avec Return). Maintenant, si le fichier est bien lu – et ses données dûment chargées dans le tableau MaxiMini, on va pouvoir continuer normalement. On va donc lire le fichier Données.txt, et à chaque ligne, on va extirper l'âge de la personne, la comparer avec le Maxi et le Mini du tableau MaxiMini[xx], et si l'âge est bien compris entre les deux bornes, on va l'afficher dans la listBox lstBoxNoms, et incrémenter de 1 le nombre de résultat(s) trouvé(s), et afficher dans la box en-dessous, lblCommentaire.

Donc, dans le cmdSelection, on en est à l'endroit où on va lire la première ligne du fichier. On va enregistrer cette ligne dans une variable locale « Ligne », déclarée comme String en début de cette procédure. De là, on va appeler une fonction, déclarée comme :

PRIVATE SUB Selection(MaLigne AS String)

Comme on le voit la procédure appelante va appeler par la commande :

Selection(Ligne)

On passe donc bien la variable Ligne à la fonction Sélection, et la valeur de Ligne est récupérée pour traitement par la variable locale « MaLigne ».

Ensuite et pour la variable MaLigne, qui comporte donc une valeur de type :

« Nom : William - Age : 33 ans »

on va extraire le « 33 », comme dit précédemment.

On a vu plus haut qu'il fallait en fait 3 opérations pour cette extraction. On va donc profiter de la priorité des opérateur en Gambas, pour tout mettre en une seule formule, ce qui est logique quand on sait que dans une opération complexe, les opérations effectuées en premier sont celles les plus en amont (elles sont d'ailleurs souvent entre parenthèses). Ainsi, en mettant la formule suivante :

SonAge = Val(Left(Right$(MaLigne, 6), 2))

on obtient la donnée utile dans une variable locale de type Integer, déclarée en début de fonction.

Note : J'ai appelé « Fonction » et non pas procédure cette Sub, en fait, je ne suis pas du tout certain

que ce soit la bonne terminologie. Normalement, une fonction renvoie une valeur, ici, elle en reçoit une, mais ne renvoie rien. Les spécialistes pourront me corriger !

Donc et pour continuer, une fois l'âge obtenu, il reste à le comparer aux Maxi et Mini, toujours en mémoire dans le tableau MaxiMini[x,x], la ligne concernée étant celle désignée par la bouton radio actuellement enfoncé qui donne dès son clic, et le nom du pays et le numéro : on a donc bien tous les éléments nécessaires pour effectuer l'opération.

La sélection s'obtient par une seule ligne :

IF SonAge <= MaxiMini[NumPays, 0] AND SonAge >= MaxiMini[NumPays, 1] THEN lstBoxNoms.Add(MaLigne)

(Attention, tout sur une seule ligne !)

Comme on le voit, on compare avec l'âge maxi (inférieur ou égal), et l'âge mini (supérieur ou égal). Si la condition est satisfaite, le candidat correspond, on va donc afficher la ligne (complète et toujours en mémoire dans « MaLigne » !), dans la listBox.

Ensuite, et pour parfaire un peu cette recherche, on va afficher le nombre de recherches trouvées, ce qui ne sera d'aucune difficulté en utilisant la propriété listBox.count.

Pour ce qui est de vider la listBox, même chose que sur la feuille Saisie, un bouton, et la commande Clear fera l'affaire. Il faut aussi noter que le fait de changer de pays avec les boutons radio devra effacer tout ce qui est affiché, tant la listBox que dans les labels au-dessus et au-dessous.

Au passage, je voudrais signaler que le fait d'afficher le pays ou encore d'afficher le nombre des résultats trouvés va devoir utiliser la langue en vigueur propre aux utilisateurs du programme. Ici et en l'occurrence, il s'agit du français. Tout le monde sait que le langage de programmation doit obéir à des règles très strictes sous peine de dysfonctionnement. Ce qui veut dire que les gens qui s'essaient à la programmation sont normalement disposés à accepter une certaine rigueur. Dans ces conditions, il est donc normal de pousser la logique dans sa pureté originelle, aussi, les messages publiés dans l'application auront à gagner en crédibilité s'ils sont affichés dans un français correct, même si cela conduit à devoir écrire une ou deux lignes de code supplémentaire, généralement fort peu contraignantes en vérité, il suffit de faire un copier / coller et de modifier une ou deux lettres. Certes, on peut se contenter d'un affichage du genre : « x résultat(s) trouvés », il faut quand même bien reconnaître que ça fait « petit joueur » et écrire une ligne avec « Un seul résultat trouvé », et une autre avec : « x & " résultats trouvés", ça a quand même une autre classe... Ah oui, bien entendu, il faut au moins un If Then avant !

Voilà, je vais m'arrêter là pour cette seconde partie.

Pour m'amuser, j'ai aussi mis un peu d'animation sur la feuille d'Accueil. J'ai ajouté une progressbar, puis un timer pour la faire évoluer. J'incrémente une variable de 0 à 100, et qui commande la progressbar jusqu'à son maximum, puis j'inverse, et décrémente de 100 à 0 pour faire redescendre. Rien d'original, c'était juste pour utiliser ce contrôle une seconde fois ! Naturellement, dès que je quitte la feuille Accueil, j'arrête le timer, pas la peine qu'il tourne si on ne le voit plus !

Toujours dans le but d'ajouter quelques futilités, je me suis dit que ce serait bien aussi d'utiliser davantage le son, aussi, j'ai mis la procédure son dans le module General, et j'ai modifié le code dans la procédure de l'affichage dans la feuille Saisies. L'appel de la procédure son se fait par son nom, en prenant soin de préciser le fichier.vaw à lire. J'ai donc deux lignes préciser, la première pour préciser exactement de quel son il s'agit, la seconde ligne pour appeler la procédure.

Ce qui donne ceci :

```
AuSon = NEW Sound(Application.path & "/LesSons/Chimes.wav") 'Chimes
General.JouerSon(AuSon)
```

En vérité, ce mode opératoire est idiot dans le cas présent : en effet, la procédure Plays dans le module ne comporte qu'une seule ligne, et il faut une ligne pour l'appeler ! Il n'y a donc dans ce cas qu'un intérêt « pédagogique »...

Dans le même temps, j'ai aussi rajouté un son à chaque occurrence trouvée dans les sélections. J'ai prévu, un son à chaque ligne, cela dit, ça va tellement vite, que je n'entends qu'un seul son ! Normalement, il faudrait exécuter cela à la fin de la recherche.

Par ailleurs, l'affichage des drapeaux à chaque click sur le bouton radio correspondant donne un affichage immédiat, donc, d'une grande banalité. Pour corser un peu, j'ai mis en place un système d'affichage progressif. Comment cela se passe ? En fait, l'astuce consiste à mettre une seconde pictureBox par-dessus la première, celle qui affiche les drapeaux, justement. Ensuite, et dès le click sur un bouton, on lance un Timer qui va modifier la taille de cette pictureBox qui ne sert que de cache, et comme le drapeau a été affiché instantanément mais occulté, il donnera l'impression de se dévoiler progressivement.

Cela n'est pas très compliqué, mais demande un peu d'attention. J'ai choisi 5 type de dévoilement : depuis la droite, depuis la gauche, depuis le haut, depuis le bas, et enfin, depuis les 4 coins jusqu'au centre. Une fois la pictureBox réduite à deux points, elle est tout simplement effacée. Attention encore : il faut que, dès le click sur un bouton de sélcetion d'un pays, que la pictureCache soit à nouveau visible, et affichée avec sa taille d'origine. Tout le code relatif à ce processus, est contenu, d'une part dans la procédure « AfficheRadio », et d'autre part, dans le TimerDrapo. Et comme ça ne suffisait pas, je cherche, dès le click de sélection, un nombre aléatoire compris entre 0 et 5, de manière d'afficher un dévoilement complètement au hasard.

Encore un mot à propos des formulaires ou feuilles : on peut bien entendu mettre une image en fond de feuille, les composants se placent bien au-dessus. Et cela donne une certaine esthétique. Attention cependant : la feuille n'a pas de propriété « Strectch » ce qui veut dire que l'image ne s'adaptera pas à la feuille ! En conséquence, il faut prévoir une image aux cotes exactes de la feuille, et interdire tout redimensionnement de celle-ci si l'on veut un affichage correct !

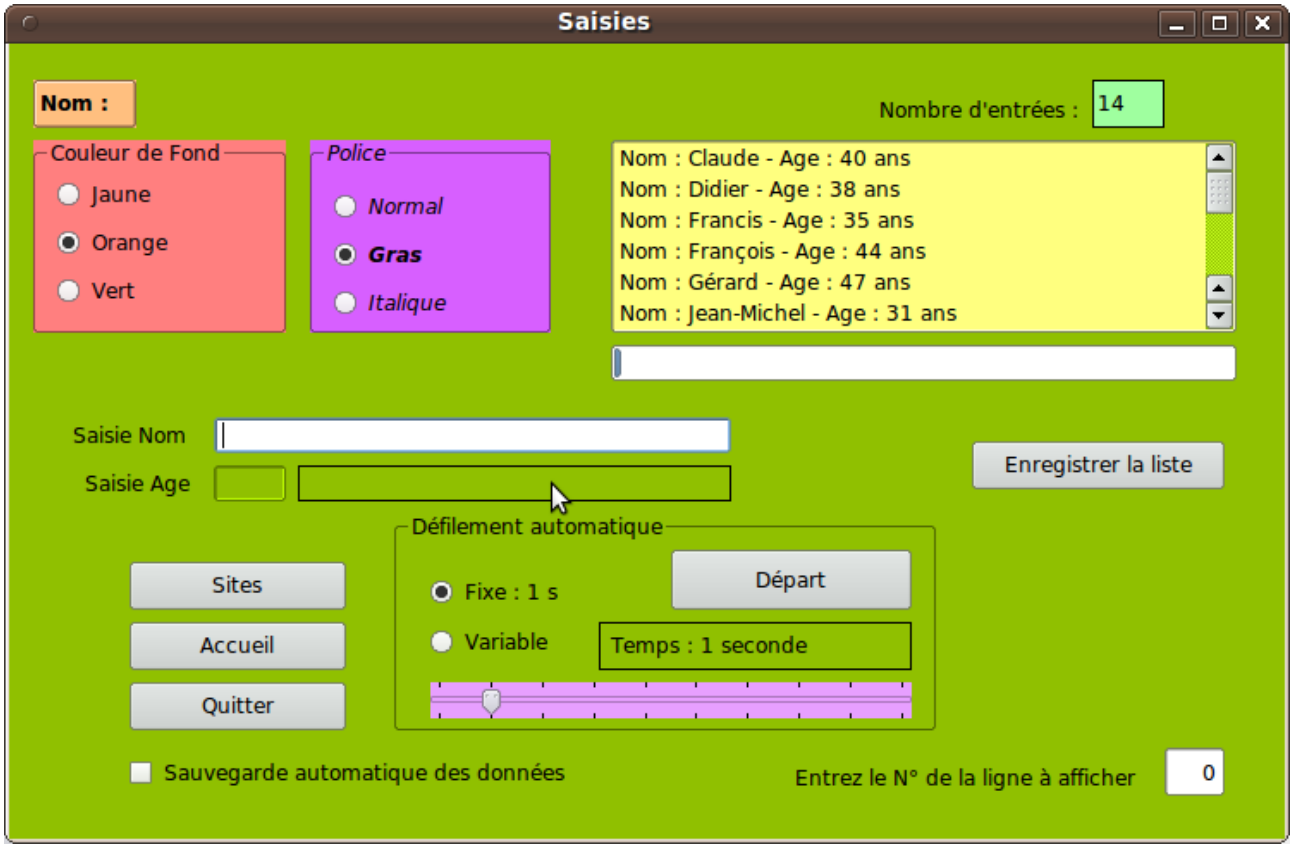
Voilà, et normalement, avec les explications ci-dessus exposées, on devrait pouvoir faire la même chose que ce que j'ai fait. Mais il faut bien considérer que c'est une étude, et que les parties de codes rappelées dans les chapitres précédents, correspondent au moment précis où elles ont été écrites. Ainsi, le code final n'est plus exactement le même que les parties mentionnées en cours.

C'est pourquoi j'invite le lecteur à éviter de regarder sérieusement le code final ci-dessous présenté, que vraiment en cas de nécessité. Il m'apparaît très clairement que la simple lecture de mon texte peut sembler quelque peu rébarbative, mais comme dit, et plusieurs fois, c'est en codant et en cherchant un peu qu'on découvre et qu'on apprend. J'ai un peu orienté mon étude comme potentiellement destinée aux anciens de Visual Basic, si tel est le cas, ils s'en sortiront facilement.

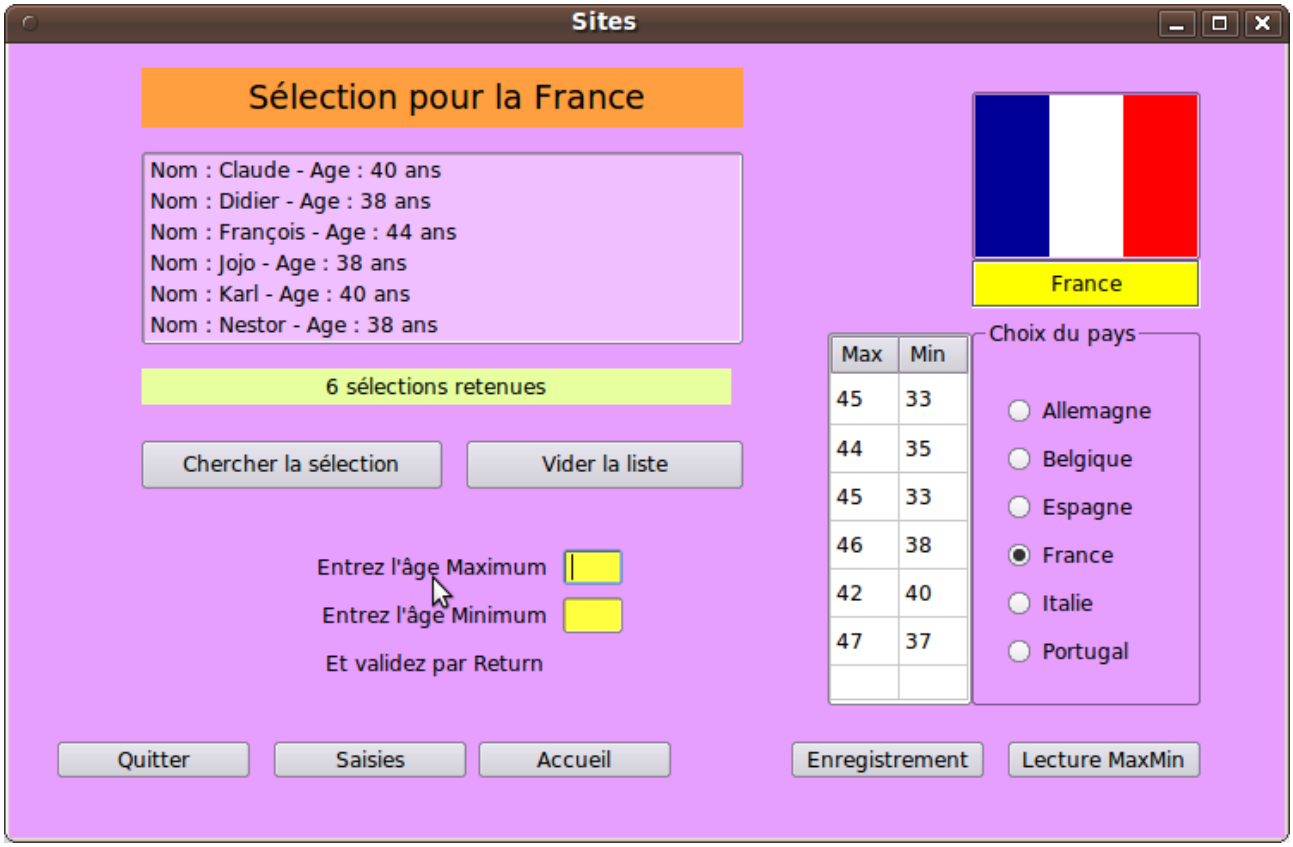
Pour info, voici les copies d'écran des 3 feuilles, telles qu'elles apparaissent à la fin de cette étude :

D'abord, la feuille d'Accueil :

La feuille Saisies :



Et enfin, la feuille Sites :



Par ailleurs, je porte ici le code utilisé. Comme déjà dit, il est perfectible, il a juste le mérite de fonctionner !

Code du module General :

```
' Gambas module file
PUBLIC SUB Fermer()
  Saisies.Quitter
  Accueil.Close
  Saisies.Close
  Sites.Close
  QUIT
END

PUBLIC SUB JouerSon(MySon AS Sound)
  MySon.Play
END
```

Code de la feuille Accueil :

```
' Gambas class file
PRIVATE Contebar AS Integer
PRIVATE FlagContebar AS Boolean

PUBLIC SUB Form_Open()
  ME.Center
  TimerDebut.Enabled = TRUE
   Panel1.X = (Desktop.Width - Panel1.Width) / 2
   Panel1.y = (Desktop.Height - Panel1.Height) / 2
END

PUBLIC SUB Form_Resize()
    Panel1.X = (Accueil.Width - Panel1.Width) / 2
    Panel1.y = (Accueil.Height - Panel1.Height) / 2
END

PUBLIC SUB cmdSaisies_Click()
 TimerBar.Enabled = FALSE
 ProgressBar1.Visible = FALSE
 Saisies.Show
END

PUBLIC SUB cmdSites_Click()
 TimerBar.Enabled = FALSE
 ProgressBar1.Visible = FALSE
 Sites.Show
END

PUBLIC SUB cmdQuitter_Click()
  TimerBar.Enabled = FALSE
  General.Fermer
END
```

```
PUBLIC SUB TimerBar_Timer()
  IF FlagContebar THEN Contebar = Contebar + 1
  IF NOT FlagContebar THEN Contebar = Contebar - 1
  IF Contebar = 100 THEN FlagContebar = FALSE
  IF Contebar = 0 THEN FlagContebar = TRUE
  ProgressBar1.Value = Contebar / 100
END

PUBLIC SUB TimerDebut_Timer()
    TimerDebut.Enabled = FALSE
    Panel1.Visible = TRUE
    FlagContebar = TRUE
    TimerBar.Enabled = TRUE
END
```

Code de la feuille Saisies :

```
' Gambas class file
PRIVATE Age AS Integer
PRIVATE Agee AS String
PRIVATE Nom AS String
PRIVATE Nome AS String
PRIVATE Erreur AS Boolean
PRIVATE Couleur AS String
PRIVATE Polisse AS String
PRIVATE FondForm AS String
PRIVATE SaveDonnees AS Boolean
PRIVATE Modif AS Boolean
PRIVATE Defil AS Boolean
PRIVATE Conteur AS Integer
```

```
PUBLIC SUB _new()

END

PUBLIC SUB Form_Open()
  ME.Center
  Erreur = FALSE
  Modif = FALSE
  Defil = FALSE
  LireFichierParametres
  LireFichierDonnees
  ChargerData
  txtSaisieNom.Text = ""
  txtSaisieAge.Text = ""
  txtSaisieAge.Enabled = FALSE
  lblNombre.Text = lstBoxNoms.Count
  txtSaisieNom.SetFocus
END

PUBLIC SUB txtSaisieNom_Change()
  Nom = txtSaisieNom.Text
  IF Len(Nom) > 0 THEN
    Label1.Text = ""
    lblAffichage.Text = "Saisie en cours... "
    txtSaisieAge.Text = ""
  ENDIF
END

PUBLIC SUB txtSaisieNom_KeyPress()
  ' on récupère le code de la touche pressée "Key.Code"
  'que l'on compare à la constante "Key.RETURN"
  'Si il n'y a pas d'erreur, on affiche
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    IF txtSaisieNom.Text = "" THEN Erreur = TRUE
```

```
    IF txtSaisieNom.Text <> "" THEN VerifNom
    IF Erreur = FALSE THEN
      Affichage 'ex cmdAfficher_Click
      txtSaisieNom.Enabled = FALSE
      txtSaisieAge.Enabled = TRUE
      txtSaisieAge.SetFocus
    ENDIF
  ENDIF
  IF Erreur = TRUE THEN
    Label1.Text = "Veuillez entrer un nom correct, SVP..."
    txtSaisieNom.Text = ""
    Nom = ""
    Erreur = FALSE
    txtSaisieNom.SetFocus
  ENDIF
END

PRIVATE SUB VerifNom()
  'Vérifier que le nom ne porte que des lettres
  DIM N AS Integer
  FOR N = 1 TO Len(nom)
    IF Asc(Mid$(Nom, N)) > 45 AND Asc(Mid$(Nom, N)) < 64 THEN Erreur = TRUE
    IF Asc(Mid$(Nom, N)) > 32 AND Asc(Mid$(Nom, N)) < 45 THEN Erreur = TRUE
  NEXT
END

PUBLIC SUB txtSaisieAge_Change()
  txtSaisieNom.Enabled = FALSE
  'Vérifier que ce ne sont que des chiffres
  IF IsNumber(Val(txtSaisieAge.Text)) THEN Age = Val(txtSaisieAge.Text)
END

PUBLIC SUB txtSaisieAge_KeyPress()
    ' On récupère le code de la touche... Comme ci-dessus
```

```
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    IF Age > 29 AND Age < 56 THEN
      Affichage
      Label1.Text = ""
      ELSE
      Label1.Text = "Veuillez entrer un nombre correct, SVP..."
      txtSaisieAge.Text = ""
      txtSaisieAge.SetFocus
    ENDIF
  ENDIF
END

PRIVATE SUB Affichage() 'ex cmdAfficher_Click()
  IF Age < 30 THEN
    lblAffichage.Text = "Nom : " & Nom
    txtSaisieNom.Enabled = FALSE
    txtSaisieAge.Enabled = TRUE
  ENDIF
  IF Age > 29 THEN
    Agee = CStr(Age)
    lblAffichage.Text = "Nom : " & Nom & " - Age : " & Agee & " ans"
    lstBoxNoms.Add("Nom : " & Nom & " - Age : " & Agee & " ans")
    lblNombre.Text = lstBoxNoms.Count
    Modif = TRUE
    cmdEnregistrerListe.Enabled = TRUE
    IF lstBoxNoms.Count > 0 THEN
      cmdSupprimer.Visible = TRUE
      cmdVider.Visible = TRUE
      IF lstBoxNoms.Count > 2 THEN
        cmdDefilement.Enabled = TRUE
        lblNumligne.Visible = TRUE
        vlBox.Visible = TRUE
      ENDIF
    ENDIF
```

```
    IF Len(Nom) > 2 THEN Nome = Nom
    txtSaisieNom.Text = ""
    txtsaisieage.text = ""
    txtsaisieage.Enabled = FALSE
    Age = 0
    txtSaisieNom.Enabled = TRUE
    txtSaisieNom.SetFocus
  ENDIF
END

PUBLIC SUB Fond_Click()
  IF rdFond1.Value = TRUE THEN
    lblAffichage.Background = &HFFFF5F&
    Couleur = 1
  ENDIF
  IF rdFond2.Value = TRUE THEN
    lblAffichage.Background = &H00FFBF7F&
    Couleur = 2
  ENDIF
  IF rdFond3.Value = TRUE THEN
    lblAffichage.Background = &H009FFFCF&
    Couleur = 3
  ENDIF
END

PUBLIC SUB Police_Click()
  IF rdPolice1.Value = TRUE THEN
    lblAffichage.Font = Font["Normal"]
    Polisse = 1
  ENDIF
  IF rdPolice2.Value = TRUE THEN
    lblAffichage.Font = Font["Bold"]
    Polisse = 2
  ENDIF
```

```
  IF rdPolice3.Value = TRUE THEN
    lblAffichage.Font = Font["Italic"]
    Polisse = 3
  ENDIF
END

PUBLIC SUB cmdSupprimer_Click()
  'Supprimer l'élément sélectionné de la liste
  lstBoxNoms.Remove(lstBoxNoms.Index)
    IF lstBoxNoms.Count < 1 THEN
    cmdSupprimer.Visible = FALSE
    cmdVider.Visible = FALSE
    cmdEnregistrerListe.Enabled = FALSE
  ENDIF
  lblNombre.Text = lstBoxNoms.Count
  Modif = TRUE
  IF lstBoxNoms.Count < 3 THEN
    cmdDefilement.Enabled = FALSE
    lblNumligne.Visible = FALSE
    vlBox.Visible = FALSE
  ENDIF
END

PUBLIC SUB cmdVider_Click()
  lstBoxNoms.clear
  cmdSupprimer.Visible = FALSE
  cmdVider.Visible = FALSE
  lblNombre.Text = "0"
  Modif = TRUE
  cmdEnregistrerListe.Enabled = FALSE
  cmdDefilement.Enabled = FALSE
  lblNumligne.Visible = FALSE
  vlBox.Visible = FALSE
END
```

```
PUBLIC SUB lstBoxNoms_Click()
  lblAffichage.Text = lstBoxNoms.Text
  ProgressBar1.Value = lstBoxNoms.Index / (lstBoxNoms.Count - 1)
  cmdSupprimer.Visible = TRUE
  cmdVider.Visible = TRUE
END

PUBLIC SUB vlBox_KeyPress()
  DIM Nbre AS Integer
  Nbre = vlBox.Text - 1
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    'lblAffichage.Text = lstBoxNoms.List[Nbre] ' 1ère méthose : directe
    lstBoxNoms.Index = Nbre        ' 2 ème méthode : va activer lstBoxNoms_Click
    PRINT Left$(Right$((lstBoxNoms.Text), 6), 2)
  ENDIF
END



PUBLIC SUB CreerFichierParametres()
  DIM FichierData AS File
  DIM Save AS Byte
  IF SaveDonnees = TRUE THEN Save = 1
  IF SaveDonnees = FALSE THEN Save = 0
  FichierData = OPEN Application.path & "/Datas/Parametres.txt" FOR CREATE
  PRINT #FichierData, Couleur
  PRINT #FichierData, Polisse
  PRINT #FichierData, Save
  PRINT #FichierData, sldTemps.Value
  CLOSE #FichierData
 END

PUBLIC SUB LireFichierParametres()
  DIM FichierData AS File
```

```
  DIM Save AS Byte
  FichierData = OPEN Application.path & "/Datas/Parametres.txt" FOR READ
  LINE INPUT #FichierData, Couleur
  LINE INPUT #FichierData, Polisse
  LINE INPUT #FichierData, Save
  LINE INPUT #FichierData, sldTemps.Value
  CLOSE #FichierData
  IF Save = 1 THEN SaveDonnees = TRUE
  IF Save = 0 THEN SaveDonnees = FALSE
CATCH
  Message.Info(Error.Text)
 END

PUBLIC SUB cmdEnregistrerListe_Click()
  'Creer la chaîne de Donnees
  DIM N AS Integer
  DIM Aenregistrer AS String
  Aenregistrer = lstBoxNoms.List[0]
  FOR N = 1 TO lstBoxNoms.Count - 1
  Aenregistrer = Aenregistrer & Chr$(13) & lstBoxNoms.List[N]
  NEXT
  'Appeler la boîte de dialogue
  Dialog.Filter = ["*.txt", "Text Files"]
  IF Dialog.SaveFile() THEN RETURN
  File.Save(Dialog.Path, Aenregistrer)
  Modif = FALSE
CATCH
  Message.Info(Error.Text)
END

PUBLIC SUB CreerFichierDonnees()
  DIM FichierData AS File
  DIM N AS Integer
  IF lstBoxNoms.Count < 1 THEN RETURN
```

```
 FichierData = OPEN Application.path & "/Datas/Donnees.txt" FOR CREATE
 FOR N = 0 TO lstBoxNoms.Count - 1
 PRINT #FichierData, lstBoxNoms.List[N]
 NEXT
 CLOSE #FichierData
END

PUBLIC SUB LireFichierDonnees()
 DIM FichierData AS File
 DIM Ligne AS String
 FichierData = OPEN Application.path & "/Datas/Donnees.txt" FOR READ
 WHILE NOT Eof(FichierData)
 LINE INPUT #FichierData, Ligne
 lstBoxNoms.Add(Ligne)
 WEND
 CLOSE #FichierData
 cmdEnregistrerListe.Enabled = TRUE
CATCH
 Message.Info(Error.Text)
END

PUBLIC SUB ChargerData()
 SELECT CASE Couleur
   CASE 1
   rdFond1.Value = TRUE
   CASE 2
   rdFond2.Value = TRUE
   CASE 3
   rdFond3.Value = TRUE
 END SELECT
 SELECT CASE Polisse
     CASE 1
   rdPolice1.Value = TRUE
```

```
    CASE 2
   rdPolice2.Value = TRUE
    CASE 3
   rdPolice3.Value = TRUE
  END SELECT
  IF SaveDonnees = TRUE THEN chkSave.Value = TRUE
  IF SaveDonnees = FALSE THEN chkSave.Value = FALSE
  IF lstBoxNoms.Count > 2 THEN
   cmdDefilement.Enabled = TRUE
   lblNumligne.Visible = TRUE
   vlBox.Visible = TRUE
  ENDIF
END

PUBLIC SUB chkSave_Click()
  IF chkSave.Value = TRUE THEN SaveDonnees = TRUE
  IF chkSave.Value = FALSE THEN SaveDonnees = FALSE
END

PUBLIC SUB cmdDefilement_Click()
  IF Defil = FALSE THEN 'Pour afficher
   Defil = TRUE
   cmdDefilement.Text = "Stop"
   TimerDefile.Enabled = TRUE
   'Afficher le premier élément
   lblAffichage.Text = lstBoxNoms.List[0]
   RETURN
  ENDIF
  IF Defil = TRUE THEN 'Pour arrêter
   Defil = FALSE
   cmdDefilement.Text = "Départ"
   TimerDefile.Enabled = FALSE
   Conteur = 0
```

```
  ENDIF
END

PUBLIC SUB TimerDefile_Timer()
  DIM AuSon AS Sound
  Conteur = Conteur + 1
  lblAffichage.Text = lstBoxNoms.List[Conteur]
  AuSon = NEW Sound(Application.path & "/LesSons/move.wav")
  General.JouerSon(AuSon)
  ProgressBar1.Value = Conteur / (lstBoxNoms.Count - 1)
  IF Conteur = lstBoxNoms.Count - 1 THEN
    Conteur = 0
    TimerDefile.Enabled = FALSE
    cmdDefilement_Click
  ENDIF
END

PUBLIC SUB rdFixe_Click()
  TimerDefile.Delay = 1000
  ProgressBar1.Value = 0 / 100
END

PUBLIC SUB rdVariable_Click()
  TimerDefile.Delay = sldTemps.Value
  ProgressBar1.Value = 0 / 100 '(lstBoxNoms.Count - 1)
  ProgressBar1.Visible = TRUE
END

PUBLIC SUB sldTemps_Change()
  TimerDefile.Delay = sldTemps.Value
  IF sldTemps.Value < 2000 THEN lblTemps.Text = " Temps : " & (sldTemps.Value / 1000) & "
seconde"
  IF sldTemps.Value > 1999 THEN lblTemps.Text = " Temps : " & (sldTemps.Value / 1000) & "
secondes"
```

```
END

PUBLIC SUB cmdSites_Click()
  Sites.Show
END

PUBLIC SUB cmdAccueil_Click()
  Accueil.Show
END

PUBLIC SUB cmdQuitter_Click()
  General.Fermer
END

PUBLIC SUB Quitter()
   DIM Reponse AS Integer
   CreerFichierParametres
   IF SaveDonnees = TRUE THEN
    CreerFichierDonnees
    Message.Info("Données enregistrées")
    ELSE
    IF Modif = TRUE THEN
       Reponse = Message.Warning("Vos données ont été modifiées, voulez-vous les enregistrer ?
", "Enregistrer", "Ignorer")
       IF Reponse = 1 THEN
         CreerFichierDonnees
         Message.Info("Données enregistrées")
         ELSE
         Message.Info("Données non enregistrées")
       ENDIF
    ENDIF
  ENDIF


END
```

Et enfin, le code de la feuille Sites :

```
' Gambas class file
PUBLIC Pays AS String
PUBLIC NumPays AS Integer
PUBLIC AgeMax AS Integer
PUBLIC AgeMin AS Integer
PUBLIC MaxiMini AS Integer[6, 2]
PUBLIC NX AS Integer
PUBLIC NY AS Integer
PUBLIC NH AS Integer
PUBLIC NW AS Integer
PUBLIC Azar AS Integer

PUBLIC SUB Form_Open()
    ME.Center
  rdPays3_Click ' Pour mettre la France par défaut
  txtAgeMax.SetFocus
  'Construire le CridView
  GRIDVIEW1.COLUMNS.Count = 2
  GRIDVIEW1.Rows.Count = 7
  GRIDVIEW1.COLUMNS[0].Width = 40
  GRIDVIEW1.COLUMNS[1].Width = 40
  GRIDVIEW1.Rows[0].Height = 30
  GRIDVIEW1.Rows[1].Height = 28
  GRIDVIEW1.Rows[2].Height = 28
  GRIDVIEW1.Rows[3].Height = 28
  GRIDVIEW1.Rows[4].Height = 28
  GRIDVIEW1.Rows[5].Height = 28
  GRIDVIEW1.COLUMNS[0].text = "Max"
  GRIDVIEW1.COLUMNS[1].text = "Min"
END

PUBLIC SUB cmdSaisies_Click()
  Saisies.Show
```

```
END

PUBLIC SUB cmdAccueil_Click()
  Accueil.Show
END

PUBLIC SUB Quitter_Click()
   General.Fermer
END

PUBLIC SUB rdPays0_Click()
  Pays = "Allemagne"
  NumPays = 0
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/allemagne.gif"] '
  AfficheRadio
END
PUBLIC SUB rdPays1_Click()
  Pays = "Belgique"
  NumPays = 1
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/belgique.gif"] '
  AfficheRadio
END
PUBLIC SUB rdPays2_Click()
  Pays = "Espagne"
  NumPays = 2
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/espagne.gif"] '
  AfficheRadio
END
PUBLIC SUB rdPays3_Click()
  Pays = "France"
  NumPays = 3
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/france.gif"] '
  AfficheRadio
END
```

```
PUBLIC SUB rdPays4_Click()
  Pays = "Italie"
  NumPays = 4
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/italie.gif"] '
  AfficheRadio
END
PUBLIC SUB rdPays5_Click()
  Pays = "Portugal"
  NumPays = 5
  picDrapeau.Picture = Picture[Application.Path & "/Images/Drapeaux/portugal.gif"] '
  AfficheRadio
END

PRIVATE SUB AfficheRadio()
  DIM AuSon AS Sound
  NH = 0
  NW = 0
  NX = 0
  NY = 0
  picCache.Width = 133
  picCache.Height = 98
  picCache.X = 560
  picCache.Y = 28
  Azar = Int(Rnd(0, 5)) ' Recherche aléatoire dévoilement drapeau
  picCache.Visible = TRUE
  TimerDrapo.Enabled = TRUE
  lblAffichPays.Text = Pays
  lblSelect.Text = " Sélection pour un pays "
  lblCommentaire.Text = ""
  lstBoxNoms.Clear
  txtAgeMax.Text = ""
  txtAgeMin.Text = ""
  lblInforme.Text = "Et validez par Return"
  AuSon = NEW Sound(Application.path & "/LesSons/Chimes.wav") 'Chimes
```

```
  General.JouerSon(AuSon)
  txtAgeMax.SetFocus
END

PUBLIC SUB txtAgeMax_Change()
  lblInforme.Text = "Et validez par Return"
  'Vérifier que ce ne sont que des chiffres
  IF IsNumber(Val(txtAgeMax.Text)) THEN
    AgeMax = Val(txtAgeMax.Text)
    ELSE
    txtAgeMax.Text = ""
    lblInforme.Text = "Nombre incorrect"
  ENDIF
END

PUBLIC SUB txtAgeMax_KeyPress()
  ' On récupère le code de la touche... Comme ci-dessus
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    IF AgeMax > 30 AND AgeMax < 56 THEN
      'Affichage
      MaxiMini[NumPays, 0] = AgeMax
      GRIDVIEW1[NumPays, 0].text = AgeMax
      lblInforme.Text = "Entrée correcte"
      txtAgeMin.SetFocus
      ELSE
      lblInforme.Text = "Entrée incorrecte"
      txtAgeMax.Text = ""
      txtAgeMax.SetFocus
    ENDIF
  ENDIF
END

PUBLIC SUB txtAgeMin_Change()
  lblInforme.Text = "Et validez par Return"
```

```
  'Vérifier que ce ne sont que des chiffres
  IF IsNumber(Val(txtAgeMin.Text)) THEN
    AgeMin = Val(txtAgeMin.Text)
    ELSE
    txtAgeMin.Text = ""
    lblInforme.Text = "Nombre incorrect"
  ENDIF
  END

PUBLIC SUB txtAgeMin_KeyPress()
  ' On récupère le code de la touche... Comme ci-dessus
  IF Key.RETURN = Key.Code OR Key.Enter = Key.Code THEN
    IF AgeMin > 29 AND AgeMin < (AgeMax - 1) THEN
      'Affichage
      MaxiMini[NumPays, 1] = AgeMin
      GRIDVIEW1[NumPays, 1].text = AgeMin
      lblInforme.Text = "Entrée correcte"
      ELSE
      lblInforme.Text = "Entrée incorrecte"
      txtAgeMin.Text = ""
      txtAgeMin.SetFocus
    ENDIF
  ENDIF
END



PUBLIC SUB cmdEnregistrer_Click()
 'Enregistrer le tableau
 DIM FichierData AS File
  DIM N AS Integer
  DIM N1 AS Integer
  FichierData = OPEN Application.path & "/Datas/AgesMaxiMini.txt" FOR CREATE
  FOR N = 0 TO 5
    FOR N1 = 0 TO 1 ' Zéro TO un !
```

```
    PRINT #FichierData, MaxiMini[N, N1]
   NEXT
  NEXT
  CLOSE #FichierData
CATCH
  Message.Info(Error.Text)
END

PUBLIC SUB cmdLecture_Click()
 DIM FichierData AS File
  DIM N AS Integer
  DIM N1 AS Integer
  lblCommentaire.Text = ""
  FichierData = OPEN Application.path & "/Datas/AgesMaxiMini.txt" FOR READ
  FOR N = 0 TO 5
   FOR N1 = 0 TO 1
    LINE INPUT #FichierData, MaxiMini[N, N1]
    GRIDVIEW1[N, N1].text = MaxiMini[N, N1]
   NEXT
  NEXT
  CLOSE #FichierData
CATCH
  Message.Info(Error.Text)
END

 PUBLIC SUB cmdSelection_Click()
  DIM FichierData AS File
  DIM Ligne AS String
  IF MaxiMini[0, 0] = 0 OR MaxiMini[0, 1] = 0 OR MaxiMini[1, 0] = 0 OR MaxiMini[1, 1] = 0
THEN
   lblCommentaire.Text = " La liste des critères n'est pas chargée ! "
   RETURN
  ENDIF
  IF NumPays = 0 OR NumPays = 2 OR NumPays = 4 THEN lblSelect.Text = " Sélection pour l'"
& Pays
```

```
  IF NumPays = 1 OR NumPays = 3 THEN lblSelect.Text = " Sélection pour la " & Pays
  IF NumPays = 5 THEN lblSelect.Text = " Sélection pour le Portugal "
  FichierData = OPEN Application.path & "/Datas/Donnees.txt" FOR READ
  WHILE NOT Eof(FichierData)
  LINE INPUT #FichierData, Ligne
  Selection(Ligne)
  WEND
  CLOSE #FichierData
 CATCH
  Message.Info(Error.Text)
 END

PRIVATE SUB Selection(MaLigne AS String)
  DIM AuSon AS Sound
  DIM SonAge AS Integer
  'Extraire l'âge
  SonAge = Val(Left(Right$(MaLigne, 6), 2))
  'Faire la sélection
  IF SonAge <= MaxiMini[NumPays, 0] AND SonAge >= MaxiMini[NumPays, 1] THEN
lstBoxNoms.Add(MaLigne)
    AuSon = NEW Sound(Application.path & "/LesSons/Ding.wav")
  General.JouerSon(AuSon)
  IF lstBoxNoms.Count = 0 THEN lblCommentaire.Text = " Aucune sélection retenue "
  IF lstBoxNoms.Count = 1 THEN lblCommentaire.Text = " Une sélection retenue "
  IF lstBoxNoms.Count > 1 THEN lblCommentaire.Text = " " & lstBoxNoms.Count & "
sélections retenues "
END

PUBLIC SUB cmdVider_Click()
  lstBoxNoms.Clear
  lblSelect.Text = " Sélection pour un pays "
  lblCommentaire.Text = ""
END

PUBLIC SUB TimerDrapo_Timer()
```

```
 ' Positions de la picture de cache X = 560 Y = 28 Haut = 98  Large = 133

 SELECT CASE Azar
  CASE 0 ' Ouvre à droite vers gauche
  NW = NW + 1
  picCache.Width = 133 - NW
  CASE 1 ' Ouvre à gauche vers droite
  NW = NW + 1
  picCache.Width = 133 - NW
  NX = NX + 1
  picCache.X = 560 + NX
  CASE 2 ' Ouvre du haut vers bas
  NY = NY + 1
  picCache.Y = 28 + NY
  NH = NH + 1
  picCache.Height = 98 - NH
  CASE 3 ' Ouvre du bas vers haut
  NY = NY + 1
  picCache.Y = 28 - NY
  NH = NH + 1
  picCache.Height = 98 - NH
  CASE 4 ' Ouvre des 4 coins vers centre
  NY = NY + 1
  NX = NX + 1
  NH = NH + 2
  NW = NW + 2
  picCache.Y = 28 + NY
  picCache.X = 560 + NX
  picCache.Height = 98 - NH
  picCache.Width = 133 - NW
  END SELECT
  IF picCache.Height < 2 THEN TimerDrapo.Enabled = FALSE
  IF picCache.Height < 2 THEN picCache.Visible = FALSE
END
```

Voilà pour le code, tel qu'il apparaît sur les feuilles.

Je propose aussi une liste de préfixes que j'utilise pour nommer mes contrôles. Elles me sont fort utiles, pour autant, elles n'ont pas vocation à devenir « références » !

A chacun de mettre les siennes !

Abréviations (personnelles) des contrôles :

Bouton OK	: **cmd**	' *Bouton de commande*	*Page : 2*
Label	: **lbl**	' *Etiquette*	*Page : 4*
TextBox	: **txt**	' *Boîte de saisie texte*	*Page : 4*
ValueBox	: **vl**	' *Boîte de saisie nombre*	*Page : 24*
ListBox	: **lst**	' *Liste*	*Page : 21*
BoutonRdio	: **rd**	' *Bouton Radio*	*Page : 11*
CheckBox	: **chk**	' *Case à cocher*	*Page : 26*
PictureBox	: **pic**	' *Image*	*Page : 44*
Slider	: **sld**	' *Curseur*	*Page : 27*
ProgressBar	: **pgb**	' *Barre de progression*	*Page : 52*
Timer	: **timer**	' *Timer*	*Page : 27*
Panel	: **pn**	' *Support sans titre*	*Page : 43*
Frame	*:* **fr**	*' Support avec titre*	*Page : 11*
TabStrip	: **tbs**	' *Support avec onglets*	*Page : N.U.*
GridView	*:* **gdv**	*'Boîte avec lignes colonnes*	*Page : 48*

Ensuite, je joins la liste des commandes utilisées, et le numéro de pages où elles apparaissent pour la première fois. Avec les modifications apportées au texte, il est possible qu'il y ait un petit décalage.

Commandes utilisées :

If / Then	*Page : 8*
Open / Read / Create	*Page : 13*
Select Case	*Page : 13*
débogage	*Page : 18*
Message	*Page : 19*
While / Wend	*Page : 25*
Warning	*Page : 26*
Enregistrer sous	*Page : 25*
CATCH	*Page : 25*
Retrurn	*Page : 28*
Sound / Delay	*Page : 28*

Voilà exposé ici ce modeste apprentissage. Le but de ce texte n'est pas de donner du code à copier / coller, c'est la raison pour laquelle il est au format pdf. Pour apprendre à coder, il faut coder, donc, se lancer, taper son texte, et en suivant ce que j'ai fait, je crois que l'on peut gagner du temps. Il est possible que certaines explications ne soient pas mentionnées, ou encore, un peu plus loin, en tous les cas, il faut se donner un peu de peine !

J'ai commencé cet apprentissage environ vers la fin de décembre 2009 soit la date de mon inscription sur le forum d'aide.

Sur ce forum, j'ai posté plus de 100 articles, je me suis fait copieusement invectiver, je comprends que les gens qui savent n'ont guère de temps à perdre avec les nigauds qui ne comprennent rien et veulent faire à leur façon. Peu m'importe, j'en ai vu d'autres, et surtout, j'ai un peu appris quelque chose. En suivant ce que je décris, le lecteur aura la possibilité d'acquérir les bases minimum, et ce, sans déranger qui que ce soit dans les forum, après, quand il se sera perfectionné, il pourra poser les bonnes questions, les plus pertinentes. Il y a aussi, je le répète une grande documentation, il ne faut pas hésiter à la consulter !

Je vous souhaite beaucoup de plaisir dans votre apprentissage et beaucoup de réalisations personnelles !

Achevé le 1 /02 / 2010