

Développement efficace des interfaces graphiques

Améliorer sa programmation sous MATLAB



par Jérôme Briot ([Dut sur developpez.com](#))

Date de publication : 05/09/2007


Dernière mise à jour : 17/03/2009

Cet article s'intéresse au développement des interfaces graphiques (GUI) sous MATLAB sans l'outil GUIDE.

Public visé : cet article ne s'adresse pas à un public débutant en programmation MATLAB mais à un public plus avancé qui maîtrise déjà la syntaxe du langage MATLAB.

Contenu : cet article présente, à partir d'un exemple simple, les différentes techniques de programmation des interfaces graphiques. Il se focalise particulièrement sur les différentes méthodes de gestion des variables.

Objectif : à l'issue de cet article, le lecteur pourra choisir parmi toutes les méthodes de développement des interfaces graphiques, celle qui lui convient le mieux.

 ***Votre avis et vos suggestions sur cet article m'intéressent ! Alors après votre lecture, n'hésitez pas :***

Avant propos.....	3
1 - Description de l'interface graphique.....	4
2 - Méthodes de développement.....	5
2.1 - Variables globales.....	5
2.2 - SETAPPDATA, GETAPPDATA et FINDOBJ.....	6
2.3 - GUIDATA et GUIHANLDES.....	8
2.4 - Les fonctions imbriquées (nested functions).....	9
3 - Quelle méthode choisir ?.....	12
Pour résumer.....	13
Remerciements.....	14

Avant propos

Depuis l'introduction de l'outil GUIDE dans la version 5.0 de MATLAB (1), le développement des Interfaces Graphiques (GUI) est devenu courant. Malheureusement, de nombreux développeurs se heurtent à des difficultés pour gérer à la fois les variables et les identifiants des objets graphiques, lorsqu'ils utilisent l'outil GUIDE. D'où l'inefficacité au long terme du développement des interfaces graphiques à l'aide de cet outil, comme l'auteur l'a déjà suggéré (2).

Cet article présente donc quatre méthodes de développement "à la main" palliant ces difficultés :

- l'utilisation des variables globales
- l'utilisation des fonctions SETAPPDATA, GETAPPDATA et FINDOBJ
- l'utilisation des fonctions GUIDATA et GUIHANDLES
- l'utilisation des fonctions imbriquées (nested functions)

Pour chaque méthode, le code est contenu dans un seul fichier m (suffisamment commenté) où l'on trouvera 3 fonctions :

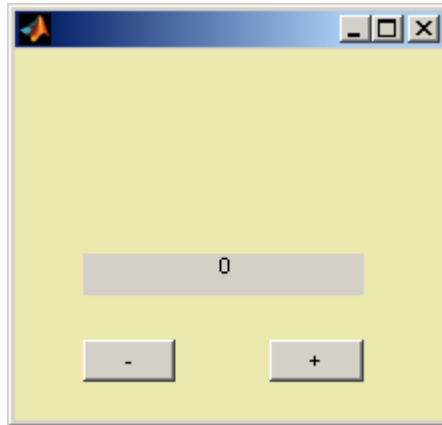
- la fonction principale (même nom que celui du fichier) qui crée les objets graphiques et initialise les variables
- une sous-fonction "retrancher" qui gère l'opération -
- une sous fonction "ajouter" qui gère l'opération +

Notes :

- cet article ne revient pas sur les différents objets graphiques disponibles déjà présentés par l'auteur (3).
- le lecteur utilisera la documentation MATLAB conjointement à cet article pour obtenir plus d'informations sur les fonctions MATLAB utilisées.
- la technique de stockage des variables dans la propriété UserData des objets graphiques n'est pas présentée ici. Malgré sa validité, il est de bon usage de laisser cette propriété disponible pour l'utilisateur de l'interface graphique (comme le nom de la propriété l'indique).


1 - Description de l'interface graphique

L'interface graphique présentée ici est un compteur très simple qui comporte une zone de texte (initialement à 0) et deux boutons (+ et -), permettant d'incrémenter le résultat (respectivement de +1 et -1).



La problématique est donc ici de gérer, d'une part l'incrémentation de la valeur courante du compteur et, d'autre part, de gérer les identifiants des deux boutons pour connaître la valeur de l'incrémentation (+1 ou -1) lorsque l'utilisateur clique sur l'un ou sur l'autre.

2 - Méthodes de développement

 *Bien lire intégralement ce tutoriel avant de choisir parmi l'une des méthodes de programmation présentées ici*

2.1 - Variables globales

Le fichier GUI_VAR_GLOBALES.M montre l'utilisation des variables globales pour gérer les variables et les identifiants des objets graphiques pendant l'exécution de l'interface graphique. Une fois les variables définies, en utilisant le mot clé `global`, dans toutes les fonctions et sous-fonctions, leur manipulation est très aisée. Elles sont constamment visibles dans toutes les fonctions. Il n'y a donc pas de gestion à proprement dit. Il suffit de les utiliser simplement sans se soucier de leur accessibilité.

Fichier gui_var_globales.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEBUT DE LA FONCTION PRINCIPALE%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function gui_var_globales

% Définition de nCompteur et handles comme variables globales dans chaque fonction et sous-fonction
% nCompteur :valeur courante du compteur (scalaire)
% handles : identifiants des objets graphiques (vecteur)
global nCompteur handles

% Initialisation de la variable représentant la valeur courante du compteur nCompteur à 0
nCompteur=0;

% Création de l'objet Figure
handles(1)=figure('units','pixels',...
    'position',[250 250 500 500],...
    'color',[0.925 0.913 0.687],...
    'numbertitle','off',...
    'name','[GUI] Utilisation des variables globales',...
    'menubar','none',...
    'tag','interface');

% Création de l'objet Uicontrol Pushbutton -
handles(2)=uicontrol('style','pushbutton',...
    'units','normalized',...
    'position',[0.1 0.1 0.1 0.05],...
    'string','- ',...
    'callback',@retrancher,...
    'tag','bouton-');

% Création de l'objet Uicontrol Pushbutton +
handles(3)=uicontrol('style','pushbutton',...
    'units','normalized',...
    'position',[0.3 0.1 0.1 0.05],...
    'string','+ ',...
    'callback',@ajouter,...
    'tag','bouton+');

% Création de l'objet Uicontrol Text résultat
handles(4)=uicontrol('style','text',...
    'units','normalized',...
    'position',[0.1 0.2 0.3 0.05],...
    'string','0',...
    'tag','resultat');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA FONCTION PRINCIPALE%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEBUT DE LA SOUS-FONCTION RETRANCHER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    
```

Fichier gui_var_globales.m

```
function retrancher(obj,event)

% Définition de nCompteur et handles comme variables globales dans chaque fonction et sous-fonction
% nCompteur : valeur courante du compteur (scalaire)
% handles : identifiants des objets graphiques (vecteur)
global nCompteur handles

% Diminution de la valeur de nCompteur
nCompteur=nCompteur-1;

% Actualisation de la propriété String de l'objet Uicontrol Text résultat
set(handles(4),'string',num2str(nCompteur));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA SOUS-FONCTION RETRANCHER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEBUT DE LA SOUS-FONCTION AJOUTER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ajouter(obj,event)

% Définition de nCompteur et handles comme variables globales dans chaque fonction et sous-fonction
% nCompteur : valeur courante du compteur (scalaire)
% handles : identifiants des objets graphiques (vecteur)
global nCompteur handles

% Augmentation de la valeur de nCompteur
nCompteur=nCompteur+1;

% Actualisation de la propriété String de l'objet Uicontrol Text résultat
set(handles(4),'string',num2str(nCompteur));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA SOUS-FONCTION AJOUTER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2.2 - SETAPPDATA, GETAPPDATA et FINDOBJ

Le fichier GUI_APPDATA_FINDOBJ.M montre l'utilisation des fonctions SETAPPDATA et GETAPPDATA pour gérer les variables et l'utilisation de la fonction FINDOBJ pour les identifiants des objets graphiques pendant l'exécution de l'interface graphique.

Fichier gui_appdata_findobj.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEBUT DE LA FONCTION PRINCIPALE%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function gui_appdata_findobj

% Création de l'objet Figure
figure('units','pixels',...
    'position',[250 250 500 500],...
    'color',[0.925 0.913 0.687],...
    'numbertitle','off',...
    'name','[GUI] Utilisation de SETAPPDATA / GETAPPDATA',...
    'menubar','none',...
    'tag','interface');

% Création de l'objet Uicontrol Pushbutton -
uicontrol('style','pushbutton',...
    'units','normalized',...
    'position',[0.1 0.1 0.1 0.05],...
    'string','- ',...
    'callback',@retrancher,...
    'tag','bouton-');
```

Fichier gui_appdata_findobj.m

```

% Création de l'objet Uicontrol Pushbutton +
uicontrol('style','pushbutton',...
    'units','normalized',...
    'position',[0.3 0.1 0.1 0.05],...
    'string','+',...
    'callback',@ajouter,...
    'tag','bouton');

% Création de l'objet Uicontrol Text résultat
uicontrol('style','text',...
    'units','normalized',...
    'position',[0.1 0.2 0.3 0.05],...
    'string','0',...
    'tag','resultat');

% Initialisation de la valeur représentant la valeur courante du compteur nCompteur à 0
nCompteur=0;

% Enregistrement direct de nCompteur dans les données d'application de l'objet Figure
setappdata(gcf,'valeur_de_nCompteur',nCompteur);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA FONCTION PRINCIPALE%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DÉBUT DE LA SOUS-FONCTION RETRANCHER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function retrancher(obj,event)

% Récupération directe de nCompteur depuis les données d'application de l'objet Figure
% contenant l'objet graphique dont l'action est exécutée (gcbf)
nCompteur=getappdata(gcbf,'valeur_de_nCompteur');

% Diminution de la valeur de nCompteur
nCompteur=nCompteur-1;

% Récupération de l'identifiant de l'objet Uicontrol Text résultat enfant de l'objet Figure
% contenant l'objet graphique dont l'action est exécutée (gcbf)
h=findobj('parent',gcbf,'style','text','tag','resultat');
% Modification de sa propriété String
set(h,'string',num2str(nCompteur));

% Enregistrement de la nouvelle valeur de nCompteur dans les données d'application de l'objet Figure
% contenant l'objet graphique dont l'action est exécutée (gcbf)
setappdata(gcbf,'valeur_de_nCompteur',nCompteur);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA SOUS-FONCTION RETRANCHER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DÉBUT DE LA SOUS-FONCTION AJOUTER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ajouter(obj,event)

% Récupération directe de nCompteur depuis les données d'application de l'objet Figure
% contenant l'objet graphique dont l'action est exécutée (gcbf)
nCompteur=getappdata(gcbf,'valeur_de_nCompteur');

nCompteur=nCompteur+1;

% Récupération de l'identifiant de l'objet Uicontrol Text résultat enfant de l'objet Figure
% contenant l'objet graphique dont l'action est exécutée (gcbf)
h=findobj('parent',gcbf,'style','text','tag','resultat');
% Modification de sa propriété String
set(h,'string',num2str(nCompteur));

% Enregistrement de la nouvelle valeur de nCompteur dans les données d'application de l'objet Figure
setappdata(gcbf,'valeur_de_nCompteur',nCompteur);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    
```

Fichier gui_apdata_findobj.m

```
%FIN DE LA SOUS-FONCTION AJOUTER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2.3 - GUIDATA et GUIHANDLES

Le fichier GUI_GUIDATA_GUIHANDLES.M montre l'utilisation des fonctions GUIDATA et GUIHANDLES. La fonction GUIDATA est utilisée pour stocker et retrouver les variables et la fonction GUIHANDLES est utilisée pour gérer les identifiants des objets

Fichier gui_guidata_guihandles.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEBUT DE LA FONCTION PRINCIPALE%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function gui_guidata_guihandles

% Création de l'objet Figure
figure('units','pixels',...
    'position',[250 250 500 500],...
    'color',[0.925 0.913 0.687],...
    'numbertitle','off',...
    'name','[GUI] Utilisation de GUIDATA',...
    'menubar','none',...
    'tag','interface');

% Création de l'objet Uicontrol Pushbutton -
uicontrol('style','pushbutton',...
    'units','normalized',...
    'position',[0.1 0.1 0.1 0.05],...
    'string','- ',...
    'callback',@retrancher,...
    'tag','bouton_retrancher');

% Création de l'objet Uicontrol Pushbutton +
uicontrol('style','pushbutton',...
    'units','normalized',...
    'position',[0.3 0.1 0.1 0.05],...
    'string','+',...
    'callback',@ajouter,...
    'tag','bouton_ajouter');

% Création de l'objet Uicontrol Text résultat
uicontrol('style','text',...
    'units','normalized',...
    'position',[0.1 0.2 0.3 0.05],...
    'string','0',...
    'tag','resultat');

% Génération de la structure contenant les identifiants des objets graphiques dont la
% propriété Tag a été utilisée.

data=guihandles(gcf);

% D'après les Tag utilisés pour les objets graphiques créés précédemment, la structure data
% contient les champs suivant :
% data.interface
% data.resultat
% data.bouton_ajouter
% data.bouton_retrancher
%

% Initialisation de la variable représentant la valeur courante du compteur nCompteur à 0
% Note : nCompteur est ici un champ de la structure data
data.nCompteur=0;

% Enregistrement de data dans les données d'application de l'objet Figure
guidata(gcf,data)
```

Fichier gui_guidata_guihandles.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA FONCTION PRINCIPALE%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEBUT DE LA SOUS-FONCTION RETRANCHER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function retrancher(obj,event)

% Récupération des données stockées dans les données d'application de l'objet Figure
% contenant l'objet graphique dont l'action est exécutée (gcbf)
data=guidata(gcbf);

% Diminution de la valeur de nCompteur
data.nCompteur=data.nCompteur-1;

% Modification de sa propriété String
set(data.resultat, 'string', num2str(data.nCompteur));

% Enregistrement des données modifiées dans les données d'application de l'objet Figure
% contenant l'objet graphique dont l'action est exécutée (gcbf)
guidata(gcbf,data)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA SOUS-FONCTION RETRANCHER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEBUT DE LA SOUS-FONCTION AJOUTER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ajouter(obj,event)

% Récupération des données stockées dans les données d'application de l'objet Figure
% contenant l'objet graphique dont l'action est exécutée (gcbf)
data=guidata(gcbf);


% Augmentation de la valeur de nCompteur
data.nCompteur=data.nCompteur+1;

% Modification de sa propriété String
set(data.resultat, 'string', num2str(data.nCompteur));

% Enregistrement des données modifiées dans les données d'application de l'objet Figure
% contenant l'objet graphique dont l'action est exécutée (gcbf)
guidata(gcbf,data)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA SOUS-FONCTION AJOUTER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    
```

2.4 - Les fonctions imbriquées (nested functions)

 Cette méthode fonctionne seulement avec la version R14 - 7.0 (ou supérieure) de MATLAB

Le fichier GUI_NESTED_FUNC.M montre l'utilisation des fonctions imbriquées (nested functions). Ce type de fonctions a été intégré dans la version 7 de MATLAB. Elles sont extrêmement souples d'utilisation. Une variable définie dans la fonction parent est uniquement visible dans les fonctions qui y sont imbriquées. L'utilisation de ces variables est donc immédiate, comme pour les variables globales.

Fichier gui_nested_func.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEBUT DE LA FONCTION PRINCIPALE%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function gui_nested_func
    
```

Fichier gui_nested_func.m

```

% Initialisation de la variable représentant la valeur courante du compteur nCompteur à 0
nCompteur=0;

% Création de l'objet Figure
handles(1)=figure('units','pixels',...
    'position',[250 250 500 500],...
    'color',[0.925 0.913 0.687],...
    'numbertitle','off',...
    'name','[GUI] Utilisation des variables globales',...
    'menubar','none',...
    'tag','interface');

% Création de l'objet Uicontrol Pushbutton -
handles(2)=uicontrol('style','pushbutton',...
    'units','normalized',...
    'position',[0.1 0.1 0.1 0.05],...
    'string','- ',...
    'callback',@retrancher,...
    'tag','bouton-');

% Création de l'objet Uicontrol Pushbutton +
handles(3)=uicontrol('style','pushbutton',...
    'units','normalized',...
    'position',[0.3 0.1 0.1 0.05],...
    'string','+',...
    'callback',@ajouter,...
    'tag','bouton+');

% Création de l'objet Uicontrol Text résultat
handles(4)=uicontrol('style','text',...
    'units','normalized',...
    'position',[0.1 0.2 0.3 0.05],...
    'string','0',...
    'tag','resultat');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEBUT DE LA FONCTION IMBRIQUEE RETRANCHER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function retrancher(obj,event)

% Diminution de la valeur de nCompteur
nCompteur=nCompteur-1;

% Actualisation de la propriété String de l'objet Uicontrol Text résultat
set(handles(4),'string',num2str(nCompteur));

    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA FONCTION IMBRIQUEE RETRANCHER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEBUT DE LA FONCTION IMBRIQUEE AJOUTER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ajouter(obj,event)

% Augmentation de la valeur de nCompteur
nCompteur=nCompteur+1;

% Actualisation de la propriété String de l'objet Uicontrol Text résultat
set(handles(4),'string',num2str(nCompteur));

    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA FONCTION IMBRIQUEE AJOUTER%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    
```

Fichier gui_nested_func.m

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FIN DE LA FONCTION PRINCIPALE%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Note : les variables utilisées à l'intérieur des fonctions imbriquées sont différentes des variables globales. En effet, ces dernières sont visibles en n'importe quel endroit du programme voire même en-dehors. Alors que la visibilité des variables des fonctions imbriquées est définie par l'ordre d'imbrication des fonctions.

3 - Quelle méthode choisir ?

L'utilisation des variables globales : bien que ce type de variable soit excessivement simple à utiliser, on tend toujours à proscrire leur utilisation. Leur totale visibilité constitue un risque majeur d'erreur de programmation si l'on ne prend pas bien soin d'assurer l'unicité de leur nom.

L'utilisation des fonctions SETAPPDATA, GETAPPDATA et FINDOBJ : cette méthode est très flexible et très fiable si le choix du nom des variables d'application est fait judicieusement. Cette méthode peut aussi être utilisée dans le cadre d'interfaces complexes à plusieurs fenêtres. Dans ce cas, il est commode de stocker les variables d'application dans l'objet graphique Root (0) pour leur assurer une visibilité totale (il faudra veiller à choisir des noms explicites dans ce cas).

L'utilisation des fonctions GUIDATA/GUIHANDLES : cette méthode est également très flexible et peut être utilisée avec du code généré par le GUIDE.

L'utilisation des fonctions imbriquées (nested functions) : cette méthode est parfaitement adaptée si le projet ne comporte qu'un seul fichier.



Un test simple pour voir la limite de l'utilisation des variables globales consiste à lancer deux fois le programme et à remarquer que les deux compteurs ne sont plus indépendants. Ce test peut être effectué sans erreur avec les trois autres méthodes.




Pour résumer

Les quatre méthodes présentées ici semblent donc utilisables dans n'importe quel projet de développement d'interfaces graphiques. Le choix dépend fortement de la complexité (multi-fenêtre) et de l'utilisation de l'interface graphique (insertion dans un projet global, maintenance,...).

Les deux méthodes les plus flexibles restent l'utilisation des fonctions GUIDATA/GUIHANDLES et l'utilisation des fonctions GETAPPDATA/SETAPPDATA. L'auteur invite donc le lecteur à choisir l'une de ces deux méthodes afin de pouvoir rapidement développer des interfaces graphiques robustes et efficaces.

Remerciements

L'auteur tient à remercier **r0d** pour la correction orthographique de cet article et **Caro-Line** pour son aide pendant la rédaction de ce même article.

- 1 :  **The Growth of MATLAB and The MathWorks over Two Decades**
- 2 :  **Introduction à la programmation des interfaces graphiques (GUI) sous MATLAB**
- 3 :  **Présentation des objets graphiques dans MATLAB**