1

Chapitre 23 : Pour aller plus loin

Table des matières

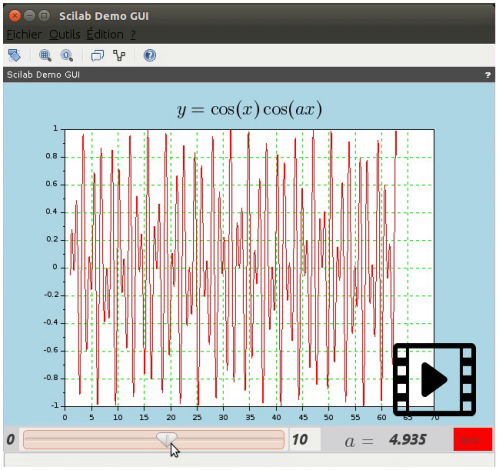
Cet extrait est avant tout destiné aux lecteurs du livre imprimée. La version 5.5 de Scilab n'a pas entraîné de changements majeurs, et les usages décrits dans livre, initialement pour la version 5.4 de Scilab, restent entièrement valables. Quelques améliorations ont été apportées sur l'interaction avec les graphiques. Parmi elles, la possibilité de rafraîchir automatiquement les éléments d'une interface graphique, qui fait l'objet d'une nouvelle section dans la version numérique du livre.

1.1. Créer ses propres interfaces graphiques (section 23.4)

Rafraîchir automatiquement les éléments d'une interface graphique

Si vous souhaitez agir de manière répétée sur une interface graphique, les éléments de l'interface doivent être mis à jour en fonction des différentes actions. Ces modifications sont effectuées par des appels répétés aux différents callbacks associés aux éléments de l'interface. Pour cela, on les place dans une boucle dont on ne sortira qu'au moment de fermer l'interface graphique. Pour expliquer ce principe nous allons prendre l'exemple d'une interface qui trace le graphe de la fonction $f(x) = \cos(x)\ \cos(ax)$ sur l'intervalle $[-10;10]$ suivant la valeur de a qu'on veut modifier à l'aide d'un curseur (slider).

Figure 1 : Graphe d'une fonction à paramètre contrôlé par un curseur



Visionnez cette vidéo sur la galerie en ligne.

L'analyse de cette interface permet de comprendre les différents éléments à programmer. D'abord l'interface graphique est représentée par un handle principal (appelons-le `G`) qui aura pour descendants les handles des autres éléments :

- un handle `A` correspondant au tracé du graphe ;
- un handle `F`, uicontrol de type *frame*, contenant les autres éléments de l'interface :
  - le handle `S`, uicontrol de type *slider*, correspondant au curseur ;
  - le handle `T`, uicontrol de type *text*, affichant la valeur du curseur ;
  - le handle `B`, uicontrol de type *pushbutton*, affichant le bouton qui ferme la fenêtre ;
  - deux autres handles correspondant aux valeurs affichées à gauche et à droite du curseur.

Parmi ces handles, les éléments qui doivent être mis à jour en fonction des événements sont au nombre de trois, ils vont correspondre aux trois callbacks à définir :

- `plot_callback` pour mettre à jour l'apparence du graphe ;
- `disp_callback` pour mettre à jour la valeur du curseur ;
- `quit_callback` chargé de fermer l'interface.

Voici maintenant le code Scilab correspondant à tous ces éléments :

```
//*****************************************
//  Mise en place de l'interface
//*****************************************

Hauteur=480;
Largeur=640;

// la figure
G = figure('position', [0 0 Largeur Hauteur],...
'Tag','figure_handle',...
'backgroundcolor', name2rgb('lightblue')/255,...
"figure_name", 'Scilab Demo GUI');

//A : Axes pour contenir le graphe
A=newaxes();
A.auto_clear="on"; // effacer automatiquement

// F va contenir les uicontrols
F = uicontrol(G,...
'Tag','frame_handle',...
'position', [0 0 Largeur-2 35], ...
'fontsize', 12, ...
'style', 'frame',...
'string', 'boutons', ...
'backgroundcolor',  name2rgb('lightgray')/255);

//le curseur
S = uicontrol('parent',F,...
'style', 'slider', ...
'Tag','slider_value',...
'position', [20 2 350 30],....
"Min",0,"Max",10,...
"value",0);

// plusieurs zones de texte
T = uicontrol('parent',F,...
'style', 'text', ...
'Tag','slider_value_display',...
'position', [500 2  80 30],...
'String',' ',...
'callback','disp_callback', ...// va afficher la valeur du curseur
'backgroundcolor',  name2rgb('lightgray')/255,...
'HorizontalAlignment', 'left', ...
'FontAngle', 'italic', ...
'FontSize', 20, ...
'FontWeight', 'bold');

T1 = uicontrol('parent',F,...
'style', 'text', ...
'Tag','slider_value_min',...
'position', [2 2  18 30],...
'String', '0',...
'backgroundcolor',  name2rgb('lightgray')/255,...
```

```
'HorizontalAlignment', 'left', ...
'FontAngle', 'italic', ...
'FontSize', 20, ...
'FontWeight', 'bold');

T2 = uicontrol('parent',F,...
'style', 'text', ...
'Tag','slider_value_max',...
'position', [372 2  40 30],...
'String', '10',...
'HorizontalAlignment', 'left', ...
'FontAngle', 'italic', ...
'FontSize', 20, ...
'FontWeight', 'bold');

T3 = uicontrol('parent',F,...
'style', 'text', ...
'Tag','var_name',...
'position', [440 2  40 20],...
'String', '$$a=$$',...
'backgroundcolor',  name2rgb('lightgray')/255,...
'HorizontalAlignment', 'right', ...
'FontSize',20);

// un bouton pour quitter l'interface
B = uicontrol('parent',F,...
'position', [Largeur-55,2 50 30], ...
'Tag','quit_bouton',...
'fontsize', 12, ...
'style', 'pushbutton',...
'string', 'Quitter', ...
'callback', 'quit_callback', ...// va stopper la boucle et fermer la
 fenêtre
'backgroundcolor', name2rgb('red')/255);


//*****************************************
//  les callbacks
//*****************************************

function disp_callback()
    //récupère la valeur du curseur
    E1=findobj("Tag","slider_value");
    value=E1.value;
    //change la valeur affichée
    E2=findobj("Tag","slider_value_display");
    E2.string=string(value)
endfunction

function plot_callback()
    //récupère la valeur du curseur
    E1=findobj("Tag","slider_value");
    value=E1.value;
    //trace la courbe
    drawlater()
    x=[1:0.001:20*%pi];
```

```
    y=cos(x).*cos(value*x)
    plot(x,y,'-r')
    xgrid(3)
    A=gca()
    A.title.text="$$y=\cos(x)\cos(a x)$$";
    A.title.font_size=4;
    drawnow()
endfunction

function quit_callback()
h=findobj("Tag","figure_handle");// récupère le handle de la fenêtre
delete(h);
abort  //  quitte la boucle while
endfunction


//*****************************************
//  lancement de l'interface
//*****************************************

cont=%t
while cont // la boucle principale
    if findobj("Tag","figure_handle")==[] then
        cont=%f; // on quitte la fenêtre prématurément
        break
    else
        F.position=[0 0 G.figure_size(1)-2 35];
       disp_callback()  // mise à jour de la valeur a
       plot_callback()  // mise à jour du graphe
       end
end
delete(gcf())  // si on quitte la fenêtre sans utiliser le boutton B
```

C'est la boucle `while` à la fin du programme qui permet d'actualiser l'interface, en fonction des actions de l'utilisateur, en appelant répétitivement `disp_callback` et `plot_callback`. L'instruction `F.position=[0 0 G.figure_size(1)-2 35];` modifie la position du uicontrol `F` contenant le curseur, les zones de texte et le bouton, en fonction de la taille de la fenêtre `G.figure_size(1)` (sa largeur en fait). Ceci permet à l'interface graphique de s'adapter aux modifications que l'utilisateur peut apporter à la taille de la fenêtre.

Astuce › *Regrouper des uicontrols dans un uicontrol de type* frame *permet de modifier facilement leur agencement dans la fenêtre puisqu'il suffit de modifier un seul handle pour modifier la position de tous ses descendants. Par contre il faudra penser le positionnement des éléments relativement au uicontrol de type* frame *et pas de manière absolue.*

Attention › *Les différentes manières de quitter l'interface graphique peuvent provoquer des erreurs irrécupérables dans la session de Scilab. En particulier, si l'interface graphique est lancée depuis le niveau d'exécution principal de Scilab alors, si la figure est fermée prématurément, on peut obtenir des messages d'erreurs du type* Le handle n'est pas ou n'est plus valide *ou même ne plus pouvoir stopper la boucle qui gère le rafraîchissement de l'interface!*