MATLAB 圖形使用者介面設計

last modified July 4, 2013

圖形使用者介面 (Graphical User Interface, GUI) 是程式語言的必備工具。主要目的在提供使用者一個互動的平台, 藉著簡易的操作便能執行該軟體 (程式) 所提供的所有功能。有別於在命令視窗進行指令式的操作, 使用門檻因此降低不少, 一般稱之爲友善的使用者介面 (Friendly User Interface)。給使用者方便, 其實是給創作者一些額外的工作 (overhead), 在原來的程式之外, 必須考慮使用者的習性與背景, 操作動線等與程式執行不直接相關的細節, 再透過另一組指令完成這件工作。本文旨在引領讀者親近 MATLAB 提供的 GUI 功能, 從做中學, 慢慢領會 GUI 設計的優勢與樂趣。

本章將學到關於程式設計

圖形使用者介面的設計觀念與 MATLAB 在這方面的技巧。

⟨本章關於 MATLAB 的指令與語法⟩
操作元 (operators):
指令: get, set
語法:

1 背景介紹

MATLAB 程式屬高階語言, 適合當做研究開發的工具, 使用者都是熟練的程式設計師。舉凡畫張常態分配的機率密度圖、直方圖等, 都是輕而易舉的事。但對於一般非程式設計人員而言, 那就不容易了, 即便只是幾行指令, 也許比登天還難。如果要以 MATLAB 程式語言設計一套工具軟體, 譬如提供使用者觀察各種分配的 pdf 與 cdf 圖, 命令視窗的指令輸入法絕非良策, 若能將所有功能包裝如圖 1 所示, 一定大受歡迎。以下練習將逐步介紹 MATLAB 在 GUI 設計上的技術。

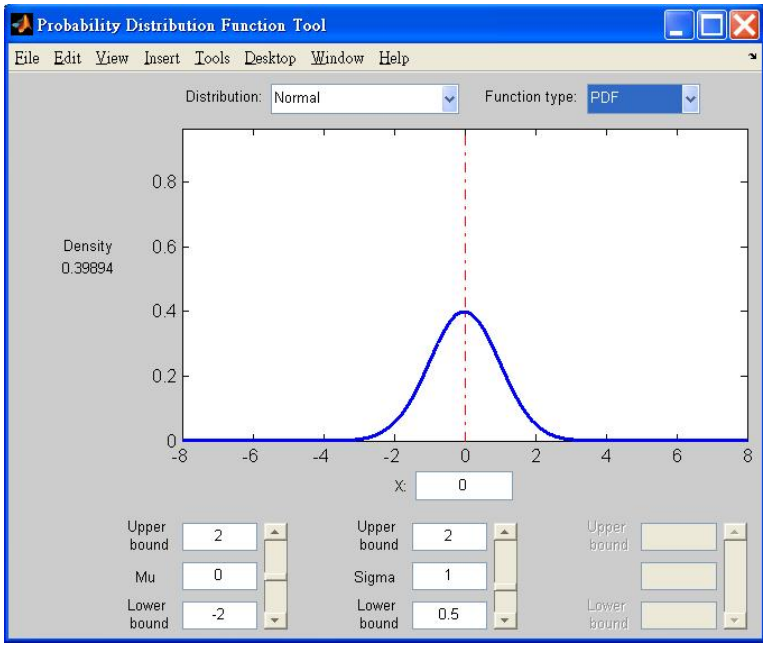


圖 1: 圖形使用者介面: 分配函數的 PDF 與 CDF 圖。

2 練習

範例 1: 凡事從簡單開始。圖 2 是一個比圖 1 簡單許多的圖形介面。想製作一個觀察常態分配函數 (pdf) 的介面, 讓使用者透過輸入不同的平均值與標準差, 觀察常態分配的長相。

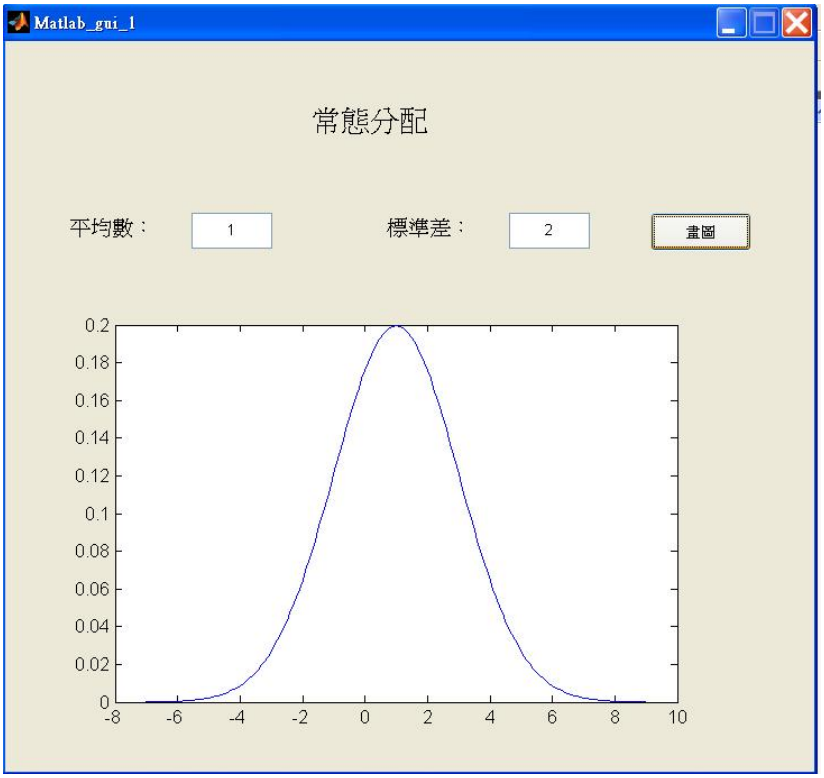


圖 2: 簡單的使用者介面程式。

GUI 程式的編輯牽涉兩個部分: 畫面的設計 (含操作的安排) 與反應程式的撰寫。一般從畫面設計編輯器 (GUIDE) 著手。新的 GUI 介面程式可以從 File − > New − > GUI 開啓。簡單如圖 2 的 GUI 介面牽涉到幾個物件: static text, edit text, push button, 及 axes。其中 static text 用來製造畫面所需的任何文字方塊, 如圖之「常態分配、平均數、標準差」。而 edit text 則是提供編輯方框, 供使用者輸入文字或數字, 如圖上白色方框內填入 1 與 2。push button 顧名思義是讓使用者按下以執行程式, 如圖上的「畫圖」按鈕。axes 是程式執行繪圖時, 呈現圖形的座標圖示。

每個 GUI 物件都是從介面環境拖曳到工作區, 如圖 3。一開始是布局的問題, 每個 GUI 物件都有幾個 property 必須被設定 (雙擊物件開啓 property 編輯器):

1. 位置、大小、文字、字體大小、顏色、預設值... 等爲數不等的 Property。

2. 配置與對齊。

3. 寫反應程式 Callback functions。例如, 按下按鈕後要執行的程式。

當 GUI 物件被拖曳到工作區時, 會被賦予一個名稱 (Tag), 在程式裡就是變數名稱, 譬如 edit1, text2 等。這個名稱在整個 GUI 裡面代表這一個物件, 當程式需要對該物件做出動作時, 必須指定正確的變數名稱。圖形使用者介面的重頭戲是, 當使用者操作該物件後所要發生的事情 (程式), 也叫做「反應程式 (callback function)」。寫在那兒呢?

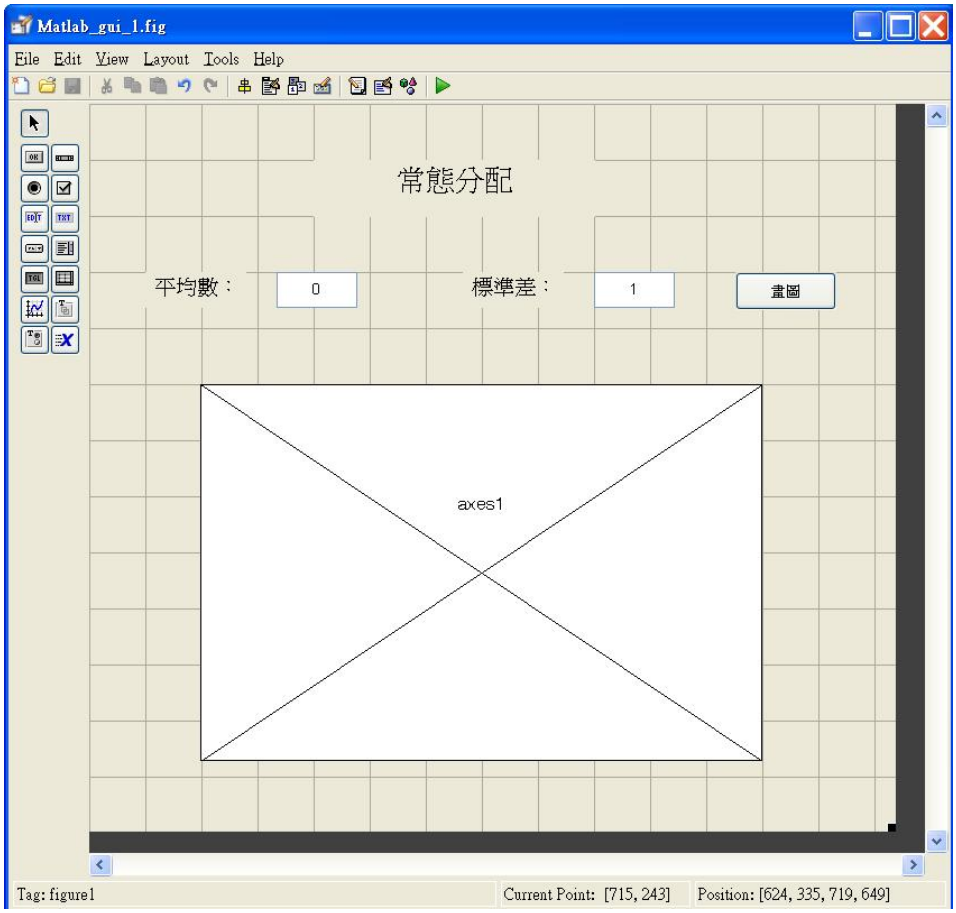


圖 3: 簡單的使用者介面程式。

當圖形介面程式第一次存檔時, MATLAB 會自動產生一個程式檔來呼應這個介面。此時有兩個檔案產生, 一個負責畫面的生成, 附檔名爲 fig , 另一個附檔名是 m , 負責處理所有的使用者介面反應。

如果設計者希望使用者在按下按鈕後執行動作, 在設計時選擇該按鈕, 按右鍵前往程式裡面負責反應的函數, 函數名稱如

```
function pushbutton1_Callback(hObject, eventdata, handles)
```

這個函數負責按下按鈕後的所有動作。典型的動作有幾個 (以圖 2 爲例):

1. get: 取得某些 GUI 物件的值, 例如使用者輸入的平均數與標準差的值。

   ```
   mu=str2double(get(handles.edit1,'string'));
   ```

   指令 get 的第一個參數 handles.edit1 表示將從 edit1 這個物件取得 string 這個 property 的內容, 隨後利用 str2double 將該字串轉換成數字。要取得畫面中某物件的內容, 必須找到代表該物件的名稱 (Tag), 在 MATLAB GUI 的設計中, 這個物件名稱「藏」在 handles 這個變數裡面, 隨著函數的呼叫傳遞。handles 是一個結構型變數 (struct), 內含所有物件的名稱。當然在物件佈置的畫面上, 也可以透過 property 的編輯, 觀察到個別物件的名稱 (Tag)。

2. set: 將執行結果寫回畫面中某個物件。set 動作與 get 相反, 讀者可以試試看。譬如,

   ```
   set(handles.text1,'string','常態分配的機率密度圖');
   ```

   看看發生什麼事了。

3. 畫圖: 利用取得的使用者輸入值, 畫出設定的圖。如果畫面中有超過一個繪圖區, 繪圖時必須指定畫到哪裡, 譬如

   ```
   plot(handles.axes1, x, f(x))
   ```

接下來, 慢慢加入其他物件, 逐漸掌握不同物件的特性。

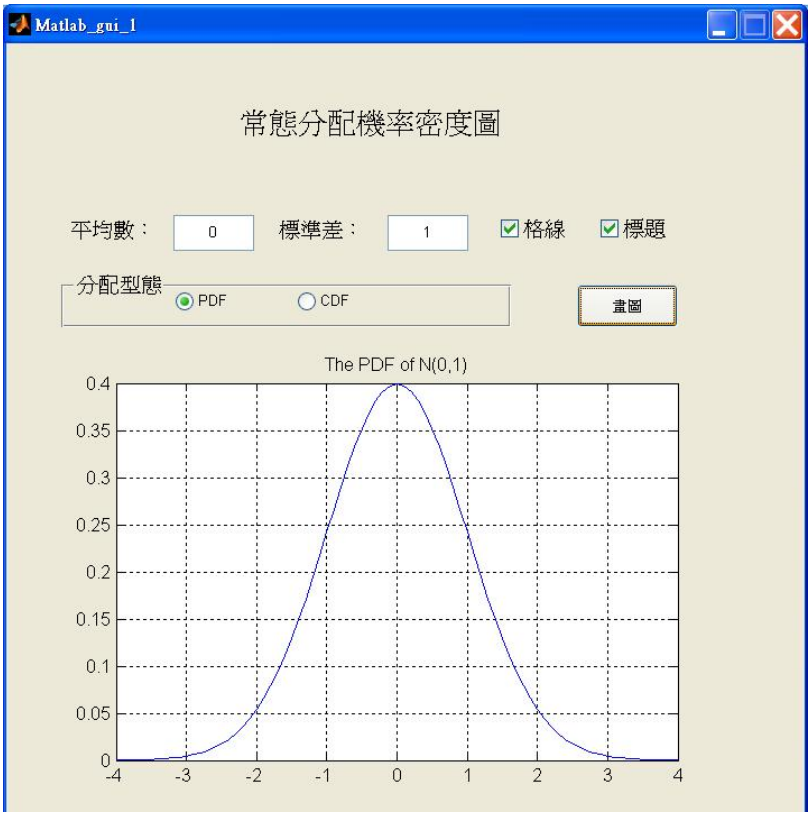


圖 4: 加入更多的使用者介面程式物件。

範例 2: 承前範例, 加入 checkbox 及 radiobutton 兩種物件, 如圖 4 所示。

---

checkbox 是一種用在多個選項當中做出多重選擇的物件 (簡稱「多選多」)。而 radiobutton 則是習慣上作爲多選項中的單選功能 (簡稱「多選一」)。這是圖形介面使用的慣例, 不可打破, 否則使用者會混淆。在 callback 程式中擷取哪個 checkbox 被打勾了比較容易, 方式如前, 譬如想知道「格線」選項是否被勾起來了,

```
g=get(handles.checkbox1,'value');
```

這個 checkbox 物件的 'value' property 非0 即 1, 如果變數 g=1 代表該選項被勾了。同樣的想知道「標題」選項是否被選了, 也如法炮製, 不再贅述。取得 g 值後, 當然必須有所反應, 譬如

```
if g==1
   grid on
end
```

至於 radiobutton 的「多選一」, 必須先群組起來, 代表在這一群選項裡面挑一個。因爲指定了群組的關係, MATLAB 會自動做出多選一的反應, 也就是同時只會有其中一個選項被選取。如果不加入群組關係, 則視爲單獨一群, 設計者自己要去處理畫面上多選一的動作。加入群組的作法, 先選取「Button Group」物件 (在左下角), 這個物件有另一個名稱叫 uipanel, 專門用來群組相關的物件, 表現方式通常是圍一個框框, 配上一個群組名 (如圖4 的「分配型態」所示)。

程式裡處理一群 radiobutton 有兩種作法; 一種如圖 4 所示, 由 pushbutton(畫圖按鈕) 壓下去後做出反應。取得哪一個 radiobutton 被選取了, 作法如下:

```
disthandle=get(handles.uipanel1,'SelectedObject');
disttype=get(disthandle, 'String')
switch disttype
case 'PDF'
   f=@(x) normpdf(x,mu,s);
case 'CDF'
  f=@(x) normcdf(x,mu,s);
end
```

第一行先取得 ButtonGroup 中哪一個物件被選了。disthandle 代表那個被選的物件的名稱 代號。利用這個代號取得該 radiobutton 的文字給變數 disttype, 再來判斷它是選項 PDF 或 CDF, 分別寫出反應動作。

另外一種處理群組 radiobutton 的方式, 是利用選取選項的同時做出反應, 因此 callback 函數由 ButtonGroup 這個 uipanel 負責, 不再交給按鈕。編輯時選取 ButtonGroup 這個 uipanel 右鍵, 選擇 View callbacks 中的 SelectionChangeFcn。此時程式會出現一個新的函數

```
function uipanel1_SelectionChangeFcn(hObject, eventdata, handles)
```

將剛才寫在畫圖按鈕的 callback 函數內的程式都搬到這裡即可。但其中 radiobutton 的選擇簡化爲

```
disttype=get(hObject, 'String')
```

大功告成。原先在圖四的「畫圖」按鈕因爲改採 radio button 的 SelectionChangeFcn 來反應使用者的選擇。因此「畫圖」按鈕不再需要, 一般改爲「關閉」按鈕 (如圖 5), 是一般圖形化介面常有的一個結束整個應用程式的按鈕。在原先處理按鈕反應的函數裡面, 加入下面幾行慣用的結束程式指令:

```
exit= questdlg('你要結束本應用程式?', 'NTPU', '是','否','否');
if exit =='是'
    close(gcf)
end
```

圖 5: 結束應用程式的詢問。

範例 3: 承前範例, 加入另一個常用的多選一物件 popupmenu, 如圖 6 上方的「選擇分配」。

Popupmenu 提供多選一的功能, 與前述的 button group 目的相同, 不同的是 popupmenu 將所有選項隱藏, 缺點是必須勞駕使用者打開才能看到所有選項, 優點是佔比較小的空間, 適用於選項衆多時。popupmenu 置入時的選項輸入方式如圖 7 所示。

圖 6: popupmenu 的使用。

本範例以「畫圖」按鈕來啓動作圖, 藉以單純化程式的內容。如前, 在程式中取得 popupmenu 選項的方式爲

```
distnameIdx=get(handles.popupmenu1, 'value')
switch distnameIdx
case 1
...
case 2
...
```

變數 distnameIdx 的值會是使用者選定的選項排序。隨後再以 switch 分流處理, 針對不同的選項做不同的動作。另外常搭配 popupmenu 使用的功能是 GUI 布局或內容的改變。例如, 本範例的 pupopmenu 提供不同分配的選擇, 但不同分配的參數名稱, 甚至數量也會不同。一般在使用者完成 popupmenu 的選擇後, 會立即反映出相對應的畫面, 如圖 8 所示。選擇「Normal」出現「平均數、標準差」等文字, 選擇「Beta」則反映出 Beta 分配慣用的參數名稱「A, B」, 而選到「Chi2」卡方分配時, 參數只有一個自由度, 於是畫面必須配合去除一組參數, 才不會讓使用者混淆。這是從事 GUI 設計比較難掌握的部份, 也就是處處替使用者著想, 不給使用者任何「犯錯」的機會, 確保整個程式操作無瑕順暢。

爲了在使用者做了選擇後立即反映出正確的畫面, 在 popupmenu 的 callback 函數必須做出以下的動作:

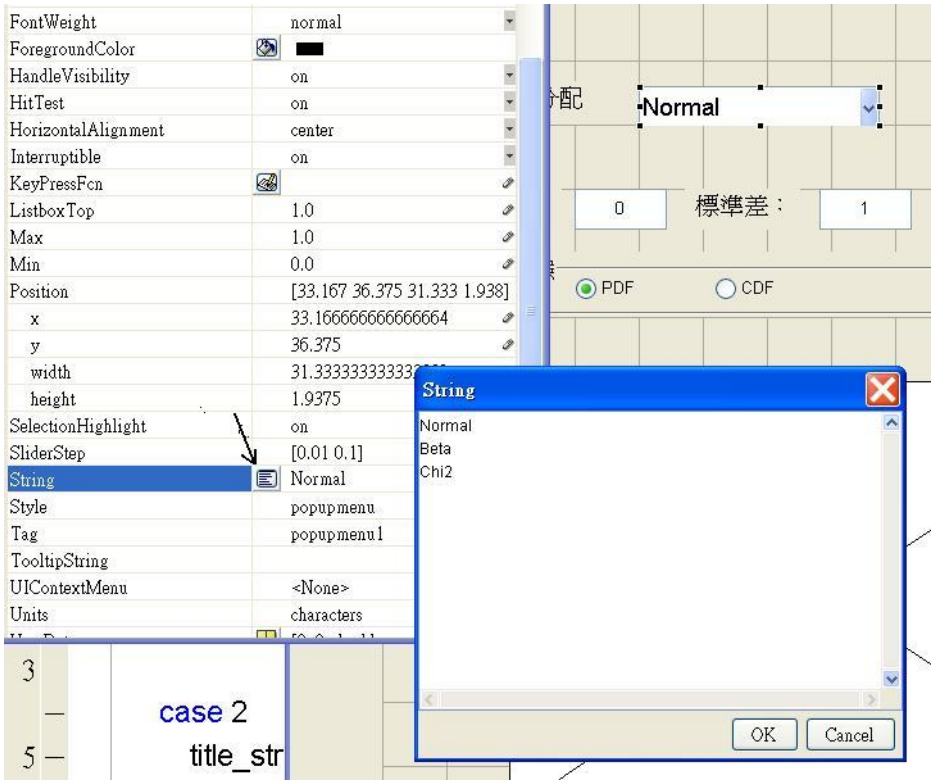


圖 7: Popupmenu 的選項輸入。

```
function popupmenu1_Callback(hObject, eventdata, handles)
distnameIdx=get(handles.popupmenu1, 'value');
switch distnameIdx
case 1
    set(handles.text2,'String','平均數');%在 text2 的位置寫入「平均數」
    set(handles.edit1,'String','0');%在 edit1 填入預設的平均數 0
    set(handles.text3,'visible','on');%恢復 text3
    set(handles.text3,'String','標準差');%在 text3 的位置寫入「標準差」
    set(handles.edit2,'visible','on');%恢復 edit2
    set(handles.edit2,'String','1');%在 edit2 填入預設的標準差 1
case 2
...
case 3
    set(handles.text2,'String','自由度');%在 text2 的位置寫入「自由度」
    set(handles.edit1,'String','2');%在 edit1 填入預設的自由度 2
    set(handles.text3,'visible','off');%隱藏 text3
    set(handles.edit2,'visible','off');%隱藏 edit2
```

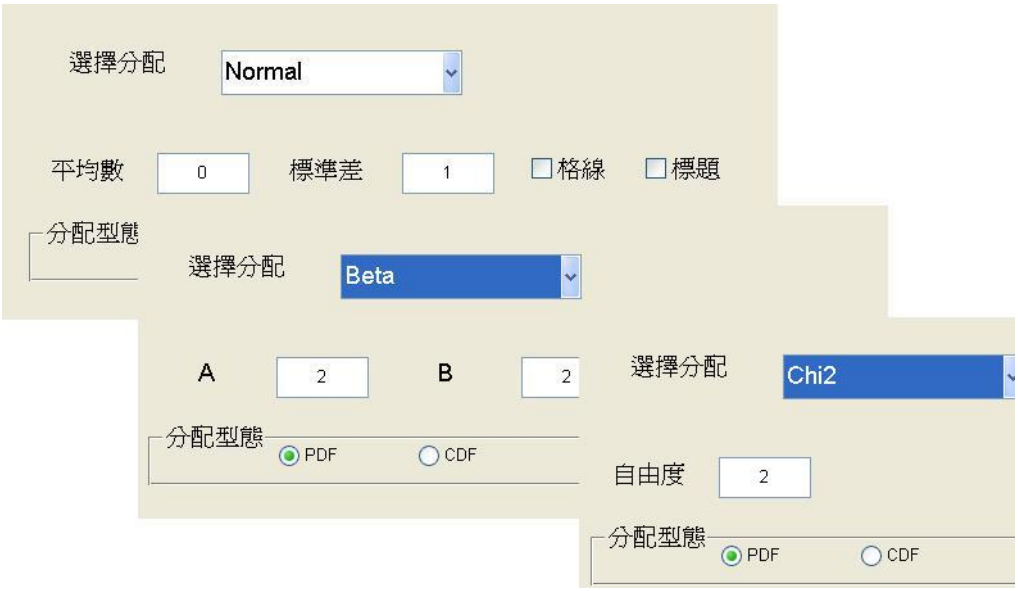


圖 8: Popupmenu 根據不同的選項變更畫面。

體貼使用者的用意雖美, 但是過程很繁瑣。從 case 1 與 case 3 的動作可以看出這些細微的小動作。例如, case 3 隱藏了 (visible off) 第二組參數, 於是在 case1 與 case 2 都必須記得還原 (visible on)。這些都不是自動發生的, 而是程式設計者必須一一去考慮與完成。case 2 與 case 1 情況相似, 留給讀者自行撰寫。

範例 4: 讀取檔案並將檔案內容顯示出來是圖形化介面常見的功能, 如圖 9 所示。這需要用到 Table 的物件及讀檔的指令。

啓動檔案的讀取通常是按鈕, 其 callback 函數主要的指令如

```
[filename, pathname] = uigetfile({'*.xls';'*.txt';'*.*'},'開啓檔案');
```

這個動作會帶出大家都熟悉的檔案選取畫面, 而 uigetfile 的第一個參數是指定讀取哪些型態的檔案, 以 cell 的資料形態表示, 一般以副檔名作爲區別, 第一順位的附檔名是預設值。第二個參數是視窗的標題。輸出結果爲檔名及所在位置的字串。接下來的程式碼當然視功能而定, 不過處理這些人機介面總有些標準動作, 用來處理一些使用者可能出現的動作, 譬如, 使用者可能沒有選取任何檔案, 直接取消動作, 這時候可以做出如下的回應:

```
if isequal(filename,0)
  msgbox('沒有選取任何檔案','File Open Error','error');
  return;
end
str=[pathname filename];
set(handles.edit1,'string',str);
```

一旦使用者選取了檔案, 程式的標準回應方式, 通常會將檔案名稱與目錄寫回畫面上。這時可以選擇寫到 static 或 edit 物件上。兩者的差別在,edit 還可以讓使用者選擇以文字輸入的方式選擇檔案, 當然這必須對 edit 的 callback 函數做出處理, 讀者可以試試看。接著便是讀取檔案並進行處理, 本範例將資料輸出在畫面的表格裡。

```
filetype = filename(end-2:end);%擷取附檔名
switch filetype
  case 'txt'
    X=load(str);
  case 'xls'
     [X, title, raw]=xlsread(str);
  otherwise
    msgbox('檔案格式錯誤, 請選擇 txt 檔或 xls 檔','File Open Error','error');
    return;
  end
```

上述指令需要對不同附檔名做處理, 因爲 MATLAB 對這讀取兩種檔案的指令不同。通常 excel 檔會含有標題列, 因此處理時可以將標題與數據分別取出 (如上述的 xlsread 指令), 放在表格的適當位置。如

```
set(handles.uitable1,'data',X);
set(handles.uitable1,'columnName',title);
```

由於 xlsread 指令已經將標題列放在 title 的 cell 變數中, 我們可以直接將這組 cell 的內容寫到表格的欄位名字上 (columnName)。第一個指令是將數據資料寫到表格的資料欄。其實表格物件的標題通常是固定的, 事先便寫好, 不從程式裡面去設定。在安排畫面時, 已經從 columnName 的選項中一一設定。讀者可以自行到 property 的 ColumnName 設定看看玩玩。倒是列名稱 (rowName) 比較常見變動的型態。讀者可以從上述對 columnName 的處理學習到如何處理 rowName。

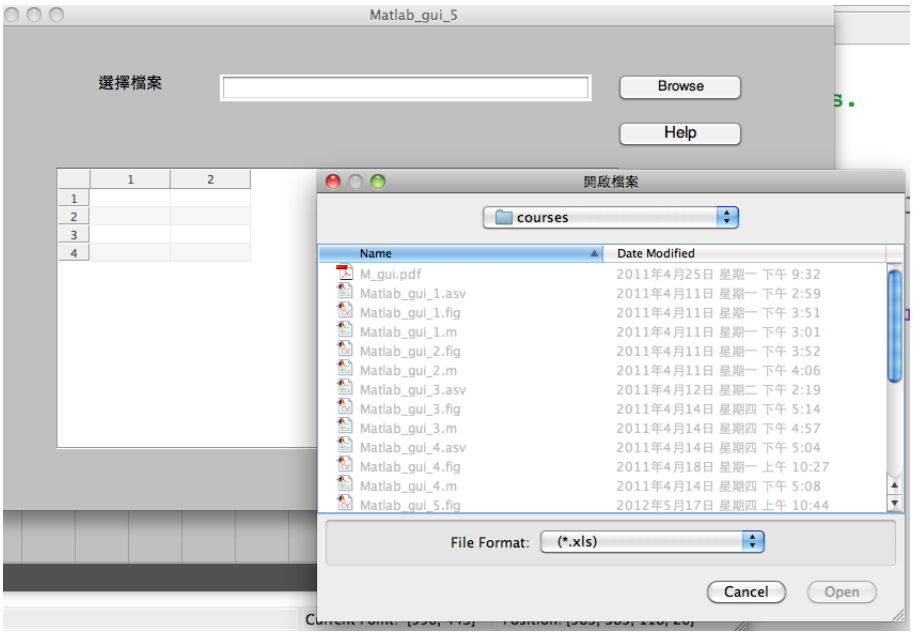


圖 9: 檔案讀取與顯示。

另外, MATLAB 也提供存檔的 GUI 介面, 讀者請自行參考 uiputfile 指令, 使用方式與 uigetfile 類似。

範例 5:GUI 的設計常會牽涉到資料共用的問題, 在多個畫面 (figure) 的情況下更是常見的技術。譬如在單一畫面下, 有共同的變數被多個物件共用。MATLAB 提供了一個很好的範例, 如圖 10 所示。這個程式做幾件簡單的事,

1. 當使用者拉動右邊的 slider, 左邊的 edit 會呈現對應的數值 (最高 1, 最低 0)。

2. 當使用者填入左邊 edit 物件數值時, 右邊的 slider 會自動調整至對應的位

置。

3. 當使用者輸入 0 至 1 以外的數值時, edit 物件會出現錯誤的訊息, 並顯示道目前爲止錯誤的次數。

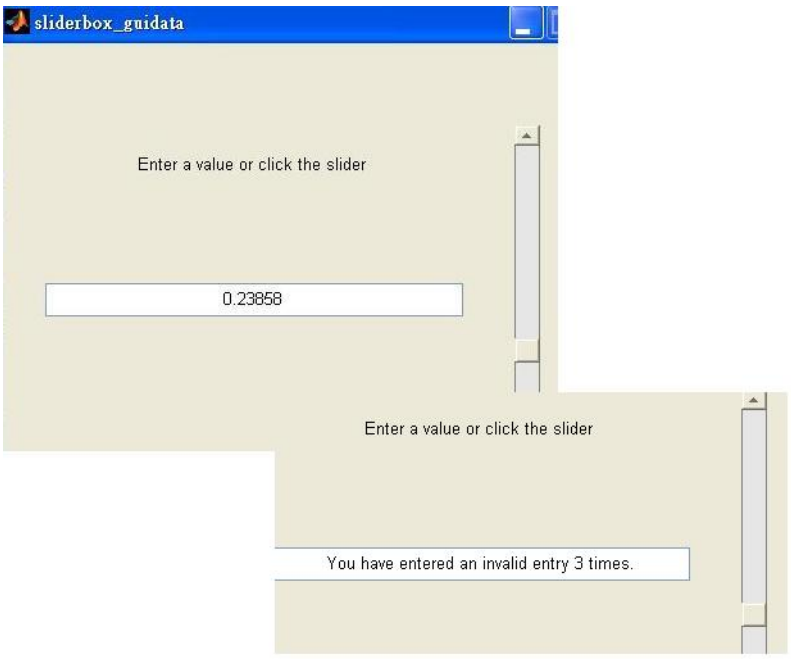


圖 10: 單一畫面的資料共用問題。

在這範例裡,「資料共用」發生在計算輸入錯誤的次數上。讀者可以先想想看, 在 GUI 的設計結構下, 如何記錄使用者操作錯誤的次數? GUI 程式由多個 function 組成,[1]共用變數的概念除了全域變數 (global variable) 之外, 還能怎麼做? MATLAB GUI 提供了幾種方式, 有興趣的讀者不妨到 Help 裡面找找看。這裡僅提出利用 GUI DATA 的方式傳遞共用資料。

所謂 GUI DATA 代表在所有的 callback 函數都可以取得或設定的資料。在前面的範例中, 就是指 handles 這個結構型的變數, 裡面包含了所有物件的代碼 (handle)。利用這個結構型變數, 我們可以擴充裡面的變數, 除了個物件的代碼外,

[1]function 內的變數只在 function 的範圍內有效。

還可以加入其他的, 譬如本範例需要的錯誤計數器。首先必須先設立計數器並預設爲 0 開始呢。下列的範例選擇在 sliderbox 被建立開啓時設定:

```
function sliderbox_guidata_OpeningFcn(hObject, ···)
···
handles.number_errors = 0;
guidata(hObject, handles);
```

在結構變數 handles 下, 新增一個變數 number_error。再利用指令 guidata 將資料存入, 才能被其他 callback 函數取得。請注意每個 callback function 都含 handles 這個結構型變數的輸入變數, handles 由系統控制並負責傳遞給被呼叫的函數。想擴充新的變數, 通常選擇在最初時被建立。
Edit 物件在它的 callback 函數用變數 handles.number_error 記錄了錯誤的次數, 圖 10 展示了適當的文字訊息, 作法如

```
handles.number_errors = handles.number_errors +1;
guidata(hObject, handles);
set(hObject, 'String',['You have ··· , num2str(handles.number_errors), 'times']
```

也就是先將計數器加 1, 再利用 guidata 存入, 最後顯示出來。
除了 guidata 外, MATLAB 也在示範程式裡建議另一種作法。[2]每個 GUI 元件都被附與一個 UserData 的 property, 用來儲存使用者資料。以本範例爲例, 可以將計數器放在 edit 這個元件的 UserData 裡。

範例 6: 前一個範例說明如何在同一個 GUI 下共用資料。本範例將前範例的 edit 與 slider 分別放置在不同的 GUI, 主程式名爲 Matlab_gui_7, 從屬程式爲 Matlab_gui_8。並介紹如何在多個 GUI 間傳遞資料。圖 11 展示兩個 GUI 間資料傳遞的示意圖。其運作如下:

[2]讀者可以在 MATLAB Help 裡輸入關鍵字「sliderbox」或「Sharing data with UserData」查到資料共用的範例, 不但可以直接執行, 也可以下載原始程式作爲參考。

1. 當使用者在 edit 填入數值後, 按「Go to Slider」開啓另一個含 slider 的 GUI (下方), 並將 edit 輸入的數值反應到 slider 上。

2. 當使用者拉動 slider 後, 按「Back to Edit」, 畫面結束並將 slider 的現值反應到原視窗的 edit 。

也就是, 在任一視窗改變的資料將反應到另一個視窗。或說在兩個視窗間傳遞資料。

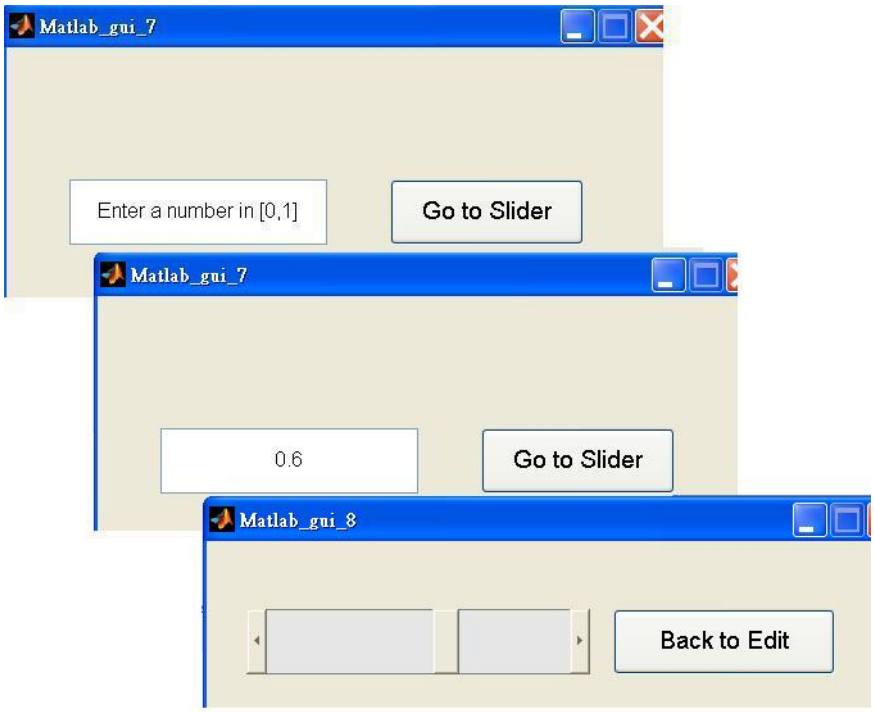


圖 11: 多個 GUI 畫面的資料共用技術。

首先分別製作兩個 GUI 程式。一個爲主程式 (Matlab_gui_7), 另一個爲從屬程式 (Matlab_gui_8)。通常從屬程式執行時, 使用者不能切換到主程式, 一來通常沒必要, 二來容易引起不必要的麻煩。所以在從屬程式裡的 WindowStyle 的 property 設爲 modal, 使用者將無法切換視窗。

本範例刻意呈現另一個常見的技術, 如圖 11 最上方的圖。先在 edit 視窗置入文字, 作爲輸入的提醒。當使用者的滑鼠移至該處並按下後, 該文字立刻消失, 留下

空白供使用者輸入數值。這些動作發生在 edit 物件的 ButtonDownFcn 的函數裡。主要動作如下:

```
set(hObject, 'string','');
set(hObject, 'Enable','on');
set(hObject,ButtonDownFcn',[]);
uicontrol(hObject);
```

另外, 在設計 edit 物件時, 必須設定 Enable 這個 property 爲 inactive。
對主程式而言, 主要動作發生在按鈕「Go to Slider」的 callback 函數。

```
current_num=str2double(get(handles.edit1,'string'));
Matlab_gui_8('Matlab_gui_7', handles.figure1,current_num);
```

第 2 行指令呼叫 Matlab_gui_8 的 GUI 畫面, 並傳遞了三個參數。第一個參數通常是一個 callback 函數, 在本範例並未使用。第二個參數傳遞了主程式的 handle, 讓從屬程式可以將資料回傳。第三個參數傳遞了目前 edit 的數值。當 GUI 被呼喚時, 通常會在 OpeningFcn 的函數中做一些初始設定。在 Matlab_gui_8 的 Matlab_gui_8_OpeningFcn 函數中, 以下的設定用來接收資料並爲交換資料做準備。

```
handles.new_number=varargin{3};
handles.mainGUI=varargin{2};
guidata(hObject, handles);
set(handles.slider,'value', handles.new_number)
uiwait(handles.figure1);
```

前兩個指令如前, 利用 handles 來儲存共享資料。最後一個指令 uiwait 令程式的執行停留在此, 直到遇到 uiresume 指令爲止。這只是方便控制 GUI 程式運作的慣用手法, 初學者先用了再來研究。從屬程式已經透過參數的傳遞得到來自主程式的資料, 但如何將從屬程式的資料傳回主程式呢? 在按鈕「Back to Edit」的 callback 函數中, 執行下列指令:

```
mGUI=guidata( handles.mainGUI);
current_num=get(handles.slider,'value');
set(mGUI.edit1,'string', current_num)
uiresume(handles.figure1);
```

第一個指令將儲存在 handles 裡面的主程式的 handle 變成眞正的 handles 結構變數, 因此可以對主程式的物件做出任何動作, 如第四行將主程式的 edit 設定爲 slider 的現值。最後的指令 uiresume 解開方才 uiwait 的等待, 直接將程式帶到輸出函數 Matlab_gui_8_OutputFcn 中, 準備離場。在這個函數的動作爲

```
varargout{1}=[ ];
delete(hObject);
```

這是標準作法, 先用了再說。如此一來, 完成了資料在兩個 GUI 間交換。主程式以輸入參數的方式傳遞資料, 從屬程式則是直接將資料設定回主程式的 edit 物件。其實, 從屬程式也可以利用上述 varargout{1}=[ ] 的方式, 改爲

```
varargout{1}=get(handles.slider,'value')
```

將 slider 的資料傳回主程式, 由主程式自行設定 edit 內容。當然原先呼叫從屬程式的指令必須改爲

```
sliderValue=Matlab_gui_8('Matlab_gui_7', handles.figure1,current_num);
```

範例 7:GUI 的設計除了應用畫布空間之外, 還可以使用一般軟體常見的「Menu Bar」, 也就是通常出現在視窗左上角的菜單, 如圖 12。這提供設計者更多選擇, 特別是一般軟體常用的「檔案」、「編輯」、「HELP」之類的選項, 可以考慮移至 Menu Bar。

加入「Menu Bar」項目的方法, 首先打開設計面板的選單「Tools – Menu Editor」, 如圖 13。「Menu Editor」可以設計兩種 menu, 一個叫「Menu Bar」也就是本範

例所指的菜單, 另一個是「Context Menus」, 將在下一個範例示範。在圖 13 Menu Bar 的設定, 主要是菜單的層次與每個選項所對應的「反應程式」。有了前面 GUI 設計的基礎, 這些選項並不難搞定, 讀者只需反覆操作幾次變可輕鬆上手。譬如, 傳統的「檔案」菜單裡面有「Open File」、「Save File」這類的選單, 讀者可以試著依照本範例的樣子, 加入其他選項與其他菜單。

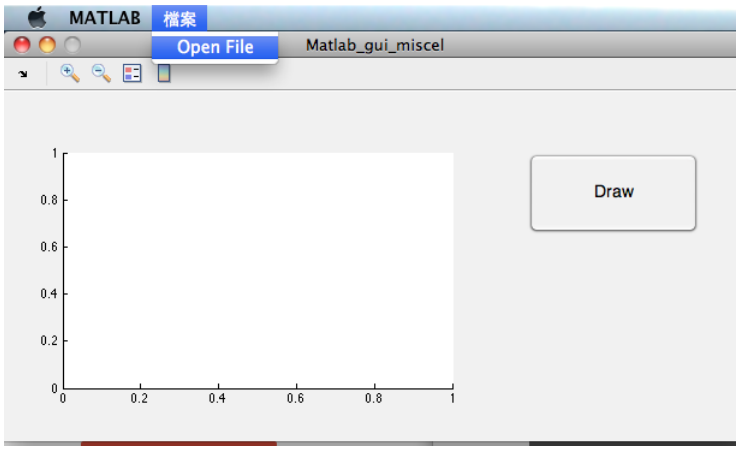


圖 12: 加入「Menu Bar」。

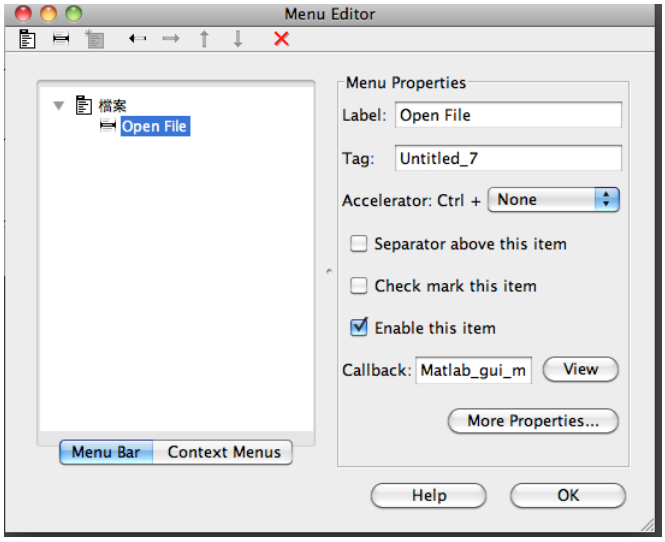


圖 13: Menu Bar 設定。

範例 8:GUI 的畫布空間隱藏另一個小世界, 如圖 14。當使用者在某個物件 (譬如,

圖形) 上點滑鼠右鍵時, 跳出來的菜單叫「Context Menus」, 提供在該物件上額外的功能選項。這個功能可以避免畫布過於擁擠的困擾, 其角色與 Menu Bar 類似。

Context Menus 的設定也是在「Tools – Menu Editor」裡面, 如圖 15。這個功能類似下拉式選單, 以幾個製作的重點; 一、當然是建立 ContextMenu, 並給予一個名稱, 如 axes_context。二、爲每個選項指定反應函數, 如圖 15(a) 的「callback」函數的指定, 這個範例則是直接在空格上寫一個簡單的指另, 將背景換成黃色。三、在目標物件 (也就是按右鍵的那個物件, 在此是 axes1) 上的 property 設定, 在 UIContextMenu 指定與那個 ContextMenu 配合。如此便大功告成。

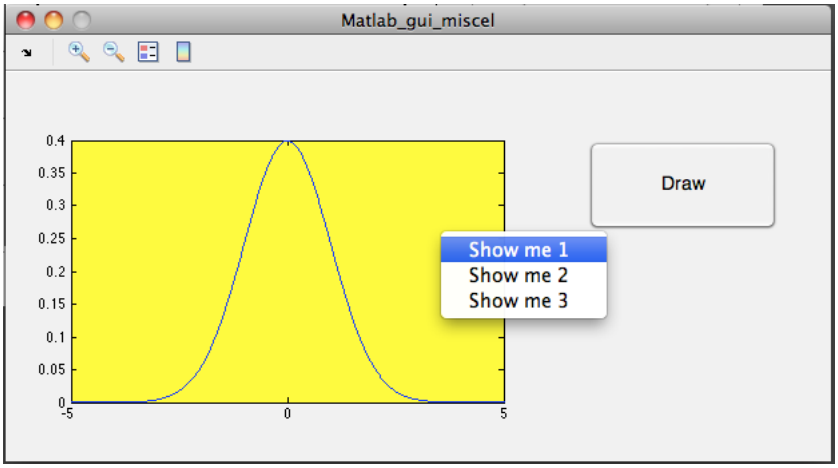
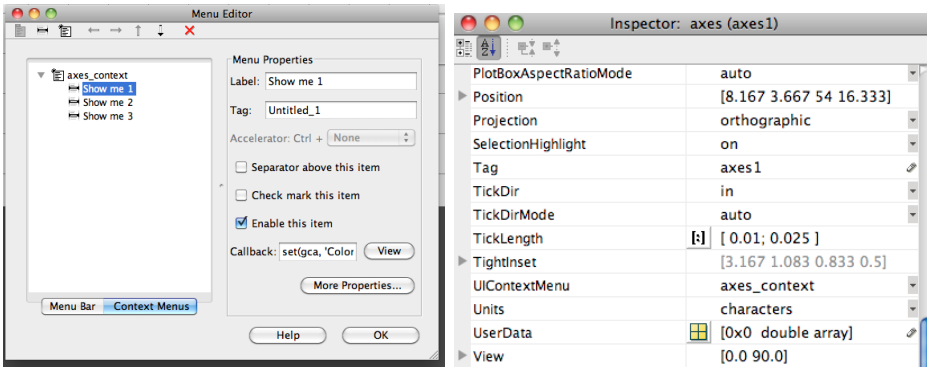


圖 14: 加入「Context Menus」。

3 觀察

1. 在一般圖形化使用者介面都會有一個「離開」的按鈕, 來結束整個畫面。這個結束程式的 callback 函數可以這樣寫:

```
exit=questdlg('關閉 xxx?','NTPUstat','YES','NO','NO');
if strcmp(exit, 'YES')
    close(gcf)
end
```



(a) 菜單與反應程式　　(b) 目標物件的設定

圖 15: Context Menu 的設定

其實關閉應用程式只要最後一行 close(gcf) 即可, 但一般處理上會加上詢問的對話盒, 目的在避免使用誤觸「離開」按鈕, 導致不可挽回的錯誤。所以無論如何, 一個好的應用程式都要有這到保險的手續, 就是惹得使用者覺得不方便都不可省略, 因爲這是設計者的責任。

2. 我們常在畫面上設計貼心的小叮嚀或短訊, 縮短學習的時間。但若有些類似手册或字數較多, 可以直接做成檔案或連結到網頁上, 使用者只要按下按鈕即可。這個 callback 函數可以這樣寫:

```
open xxx.pdf
```

這個指令會帶出瀏覽器並在裡面開啓檔案。若要連接到某個網頁 (譬如, 台北大學首頁), 改成

```
web http://www.ntpu.edu.tw/
```

3. 通常一個 GUI 程式被呼叫時, 其程式 (函數) 被呼叫的順序爲 (假設檔名爲 foo.m) foo,foo_OpeningFun, 最後是 foo_OutputFun。GUI 畫面在函數 foo 裡面被開啓。這三個函數被順序執行後, 程式結束,GUI 畫面停在螢幕上等待使用者做動作, 之後便都是執行回應程式。除非, 在 foo_OpeningFun 做出 uiwait , 迫使程式暫留, 直到在某個反應程式執行 uiresume 爲止, 程

式才跑到 foo_OutputFun, 結束程式。這個暫留動作讓程式有機會傳回資料給呼叫它的程式, 達到資料傳遞的任務。

4. 做好的 GUI 程式有時候會在不同電腦上開啓, 位避免因螢幕大小造成視窗外觀不利觀賞使用, 設計時可以加入一個選項, 使 GUI 程式在開啓後可以允許使用者調整外觀的大小。設計者只要在選單的「Tools – GUI Options」裡面的第一項,Resize bahavior, 選擇「Proportional」即可。

4 作業

1. 完成範例 1。
2. 完成範例 2, 以「畫圖」按鈕啓動畫圖。
3. 完成範例 2, 以「radio button」啓動畫圖。
4. 完成範例 3。
5. 更改範例 3 中的 Popupmenu 爲 Listbox。
6. 完成範例 4。