

Week 5-操作介面設計



GUI種類介紹

操作介面設計要求

好的操作介面讓使用者可以更直觀的完成設定，許多工程軟體在演算法的準確性及速度並無軒輊，最後卻是操作介面的設計好壞決定了銷售的成敗。對於自動化程式來說，操作介面設計決定了之後有多少人會使用，因此對於自動化程式開發者而言，操作介面的設計不可不慎。一個好的介面具備必須下列的特點：

- 簡潔的畫面
- 清楚的說明
- 正確的互動
- 當輸入值超出範圍時必須提醒使用者
- 當執行時間較長，須提供進度指示
- 本次設定為下次啟動的初始值

GUI操作介面技術

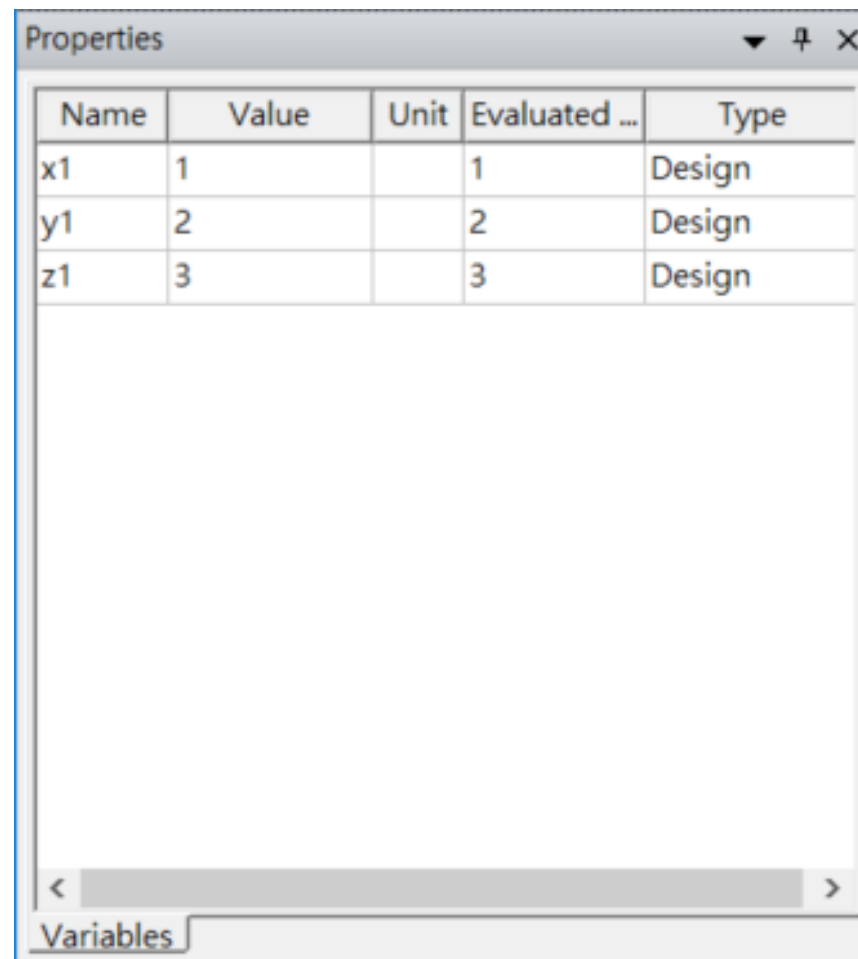
以下介紹幾種在AEDT當中常用的操作介面技術：

- 屬性設定視窗
- Tkinter
- WPF
- EXCEL & CSV
- ACT

屬性設定視窗

這個方法讓使用者可以在屬性視窗設定變量，程式再去屬性視窗讀取數值執行運算。這個方法適用於windows和linux。好處是不需要另外設計操作介面代碼，缺點是只有輸入欄位，缺乏下拉選單或是滑桿等比較進階的控鍵可以使用。

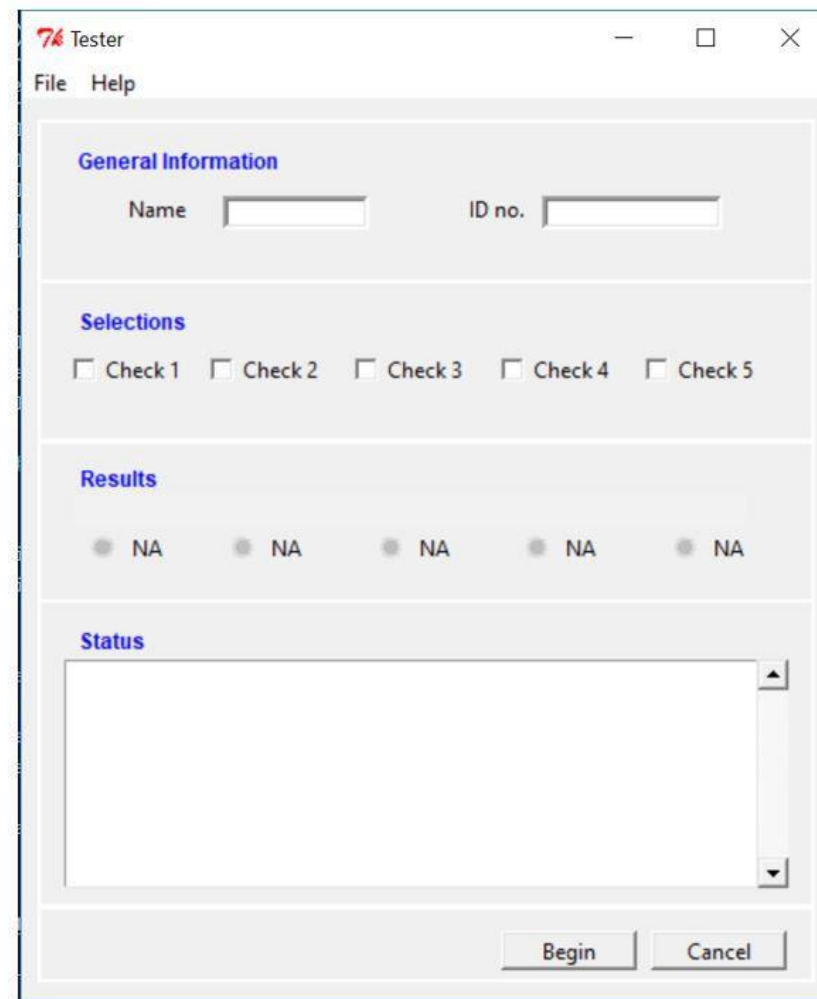
在第一次執行時程式必須加入屬性，提示使用者屬性已被加入並結束程式。只有在屬性已經存在的條件底下程式才會執行工作。屬性的命名必須具備獨特性，避免與使用者自定義變量衝突，造成程式不正常運作。建議可以加入前後底線來做區分。



Name	Value	Unit	Evaluated ...	Type
x1	1		1	Design
y1	2		2	Design
z1	3		3	Design

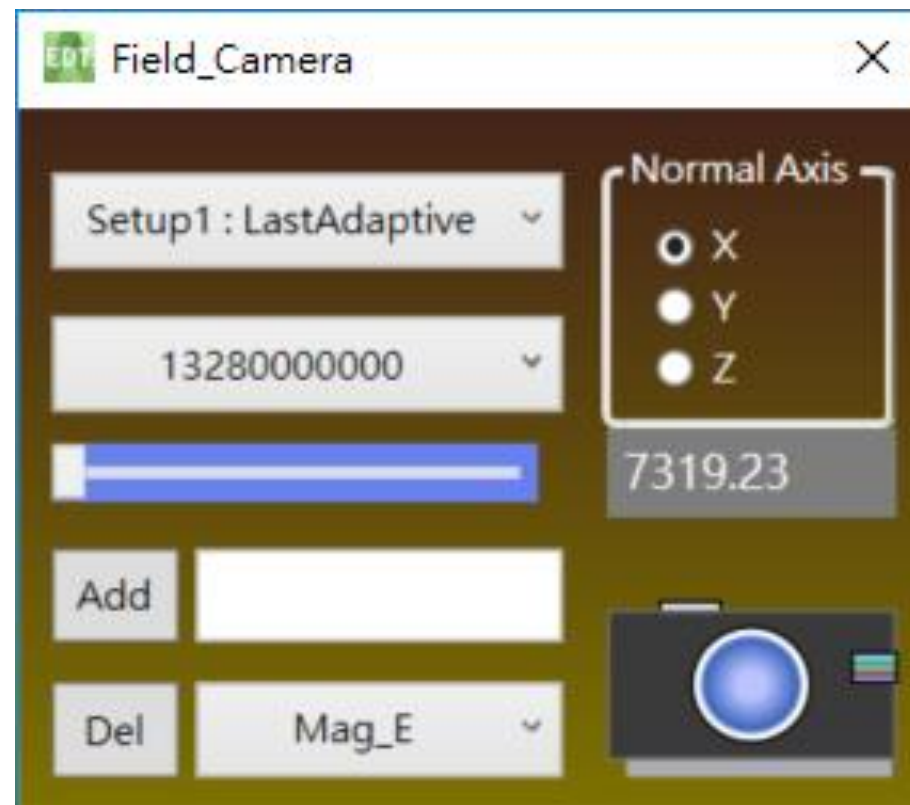
Tkinter

採用tkinter技術所建立的視窗同樣可以運行於windows和linux。然而要建立一個稍微複雜一點的視窗，就需要上百行的tkinter函式才能達成。可讀性及擴充性都不是很理想。要做到視窗控鍵的互動也需要自定義事件函數。執行起來同樣的麻煩。



用wpf的技術來開發操作視窗簡便又快速。只要拖拉控鍵到視窗之上並輸入屬性就可以輕輕鬆鬆打造出一個相當有質感的視窗。視窗定義代碼記錄在xaml當中，事件函式則是在python當中。

介面與業務代碼分離讓可讀性及擴充性大幅的提高。可惜的是現階段只能使用在windows環境當中。linux無法支援。



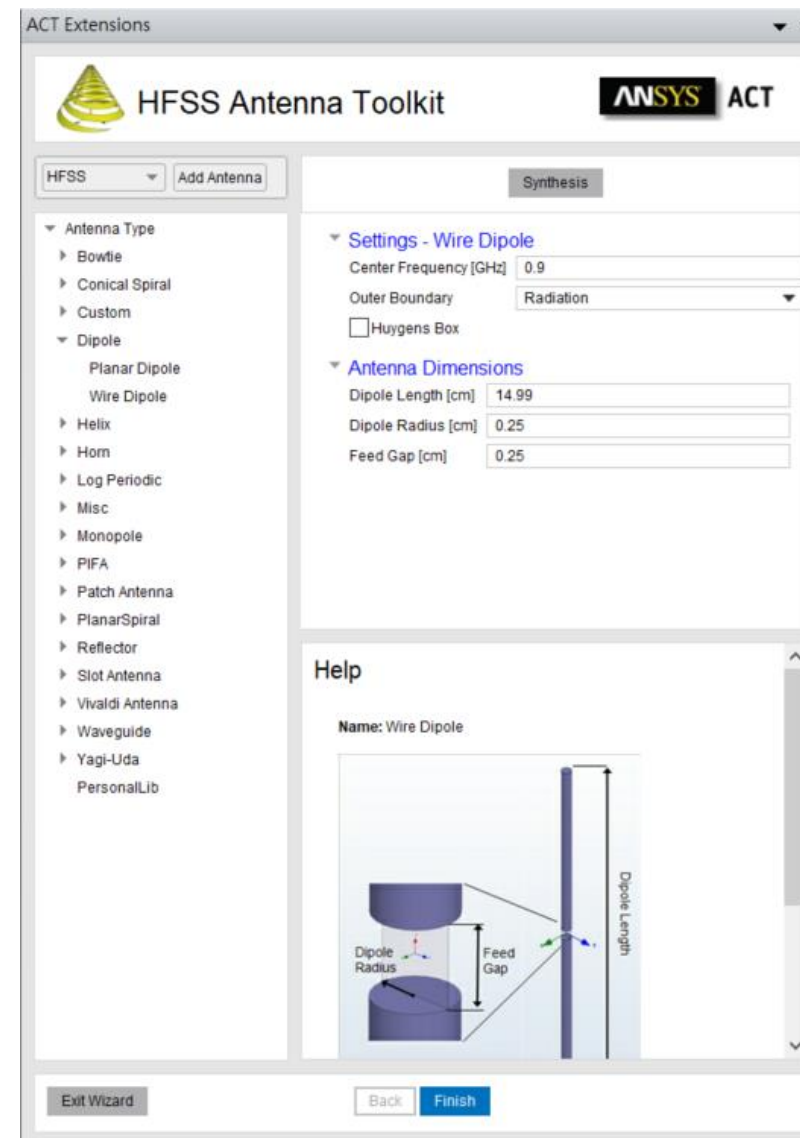
EXCEL & CSV

EXCEL作為介面，程式執行時讀取特定欄位的數值作為輸入執行運算。EXCEL可以輕鬆的做出複雜的介面，還可以加入參數檢查機制，也可以做到一定程度控鍵互動。AEDT內建函式庫提供函數直接讀取EXCEL,不需另外安裝庫。可惜的是這種方式只適用於windows環境，不適用Linux。

如果不需要華麗的介面及繁複的控鍵，最簡單的方法便是讓使用者將參數變量以txt或csv格式填寫。python讀入之後解析出參數值並執行運算即可。適用於windows和linux。可以透過OpenFileDialog()來選擇檔案

	A	B	C	D
2	General Information			
3	Project Name	test		
4	Working Directory	d:/demo2		
5	Package File (.sip)	D:/Customer2020/2020_3_19_SPIL_Automation/RRR03121136.sip		
6	Date		User	
7				
8	BGA Information			
9	Name	Ball Height (um)	Ball Radius (um)	Merge Sink
10	BGA	300	100	TRUE
11				
12	Die Information			
13	Name	Bump Height (um)	Bump Radius (um)	
14	B60874A	50	50	
15				
16				
17				
18				
19	Layers Information			
20	Name	Thickness (um)	Dk	Df
21	UNNAMED_000	50	1	0.2
22	L1	15	4	0.2
23	DRILL	80	4	0.2
24	L2	15	4	0.2
25	UNNAMED_009	50	1	0.2
26				
27				
28				
29				
30				
31				
32				

ACT可以支援幾種簡單的控鍵，如輸入欄位，下拉式選單或是標籤等等。在安裝之後並啟動之後，ACT視窗會顯示在主畫面右側，並持續保持開啟的狀態，ACT操作上相對簡便也同時支援windos和Linux。缺點是ACT屬於獨有的工具，沒有太多範例可以參考。除官方文件，網路上也找不到太多的相關資料，需仰賴使用者自行摸索。



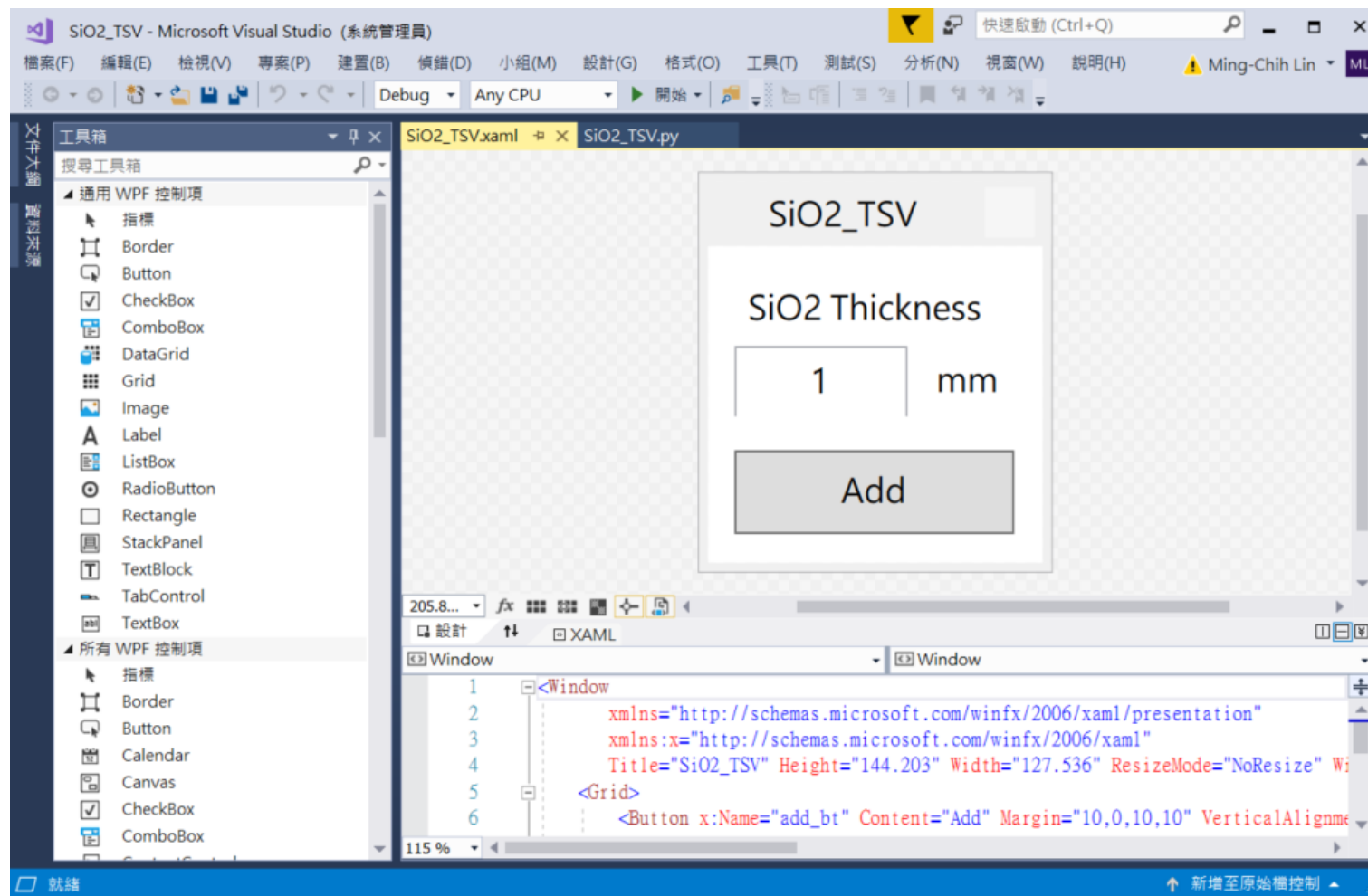
技術總結

要設計出一個好的操作介面，需要從使用者角度來思考，推測使用者輸入時可能發生的錯誤，並針對錯誤提供正確的反饋，讓使用者清楚該如何修正。建議在介面完成初步開發時便可以邀請將來可能的使用者試用，根據使用者意見適當的修改可減少未來使用上的抱怨。

從以上的介紹各位可以了解沒有哪一種技術可以涵蓋所有的好處，開發人員必須視狀況決定採用哪一種技術。

範例解說：使用Visual Studio產生GUI

Visual Studio IDE 介面



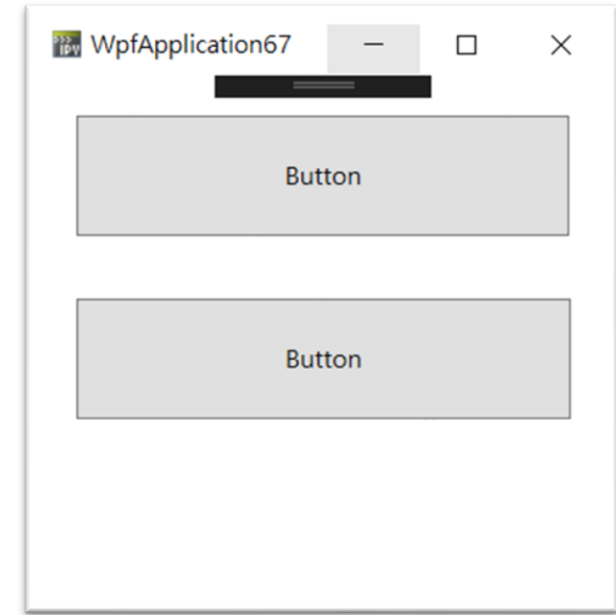
Wpf的操作介面框架

傳統的介面設計是由程式碼搭出來的。也就是透過函式呼叫來宣告視窗，按鈕，選單等控鍵，也包括了大小，位置，顏色等屬性。除了介面設計，觸發函式的定義及響應也需要編寫函式碼。因此就算是一個簡單的操作介面也動輒超過數百行代碼，不只可讀性差，也很難除錯及擴充。

新的操作介面框架則是將外觀設計與觸發函數定義在xaml檔當中，這讓設計的語法大幅的精簡。一般的介面設計師可以在不需要編程的狀況底下完成介面的外觀設計，著重美觀及流暢的使用者體驗。至於函數響應的部分則仍然由程式工程師來完成，像是查詢或是運算等等。拆分讓整個程式碼大幅的簡化，不會混在一起導致難以閱讀。

XAML代碼及對應之操作介面

- 對於自動化程式設計來說，一般介面比較單純，程式設計師可以用現成“即視即所得”的工具拖放來產生xaml碼，並同時生成觸發函數的宣告及響應的框架。在測試無誤後再行編寫響應函數的實質功能。操作介面設計設計的工作也因而大幅簡化。



```
1 <Window
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     Title="WpfApplication67" Height="300" Width="300">
5     <Grid>
6         <Button Content="Button" Height="60" Margin="20,20,20,0" VerticalAlignment="Top"
7             RenderTransformOrigin="0.571,0.499" Click="Button_Click"/>
8         <Button Content="Button" Height="60" Margin="20,111,19.6,0" VerticalAlignment="Top"
9             RenderTransformOrigin="0.571,0.499" Click="Button_Click1"/>
10    </Grid>
11</Window>
```

何謂觸發及響應

當按下按鈕或是在輸入欄位填入一個字元，都可以觸發對應的函數。也就是函式會去執行對應的工作，比方開啟對話框或是輸出建議字串。在操作介面上的任何動作都可以觸發一個事先定義好的函式。一個介面可以定義的觸發事件可能高達上百上千個。我們只要選擇需要反應的去編寫響應函數即可。簡單介面對介面的響應甚至可以在xaml當中就可以宣告，不需另外編寫函式。

```
1 <Window
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     Title="WpfApplication67" Height="300" Width="300">
5     <Grid>
6         <Button Content="Button" Height="60" Margin="20,20,20,0" VerticalAlignment="Top"
7             RenderTransformOrigin="0.571,0.499" Click="Button_Click"/>
8         <Button Content="Button" Height="60" Margin="20,111,19.6,0" VerticalAlignment="Top"
9             RenderTransformOrigin="0.571,0.499" Click="Button_Click1"/>
10    </Grid>
11</Window>
```

```
import wpf

from System.Windows import Application, Window

class MyWindow(Window):
    def __init__(self):
        wpf.LoadComponent(self, 'WpfApplication67.xaml')

    def Button_Click(self, sender, e):
        pass

    def Button_Click1(self, sender, e):
        pass

if __name__ == '__main__':
    Application().Run(MyWindow())
```

範例解說：使用屬性欄位

範例解說：使用EXCEL

專案問題解析