以 VTK 為基礎之視覺化程式設計環境在科學視覺上之應用

A Visual Programming Environment Based on VTK for Scientific Visualization

張顧耀　　廖麗雅　　盧一屏

大葉大學資訊工程學系

{canseco | R9706021 | R9606010} @mail.dyu.edu.tw

摘要

VTK（Visualization Toolkit）是一個三維電腦製圖、影像處理和科學視覺化的 C++類別函式庫。除了 C++之外，使用者亦可透過 VTK 所提供直譯式的介面，來使用該函式庫。使用者如欲利用 VTK 的各種功能，必須具備足夠的物件導向程式設計能力，或是熟悉 VTK 所支援的直譯式語言。對於臨床的醫師或一般研究人員而言，VTK 的學習門檻是十分高的。

本論文提出一個以 VTK 為基礎的視覺化程式設計環境，以解決上述學習 VTK 門檻過高的問題。該系統能讓使用者在圖形化介面中，以滑鼠直接拖拉各種 VTK 物件並加以連接的方式，建構出一個執行管線；同時，使用者亦可透過對話盒來設定 VTK 物件的屬性或參數。每一個執行管線不但能夠執行並顯示其結果，亦可輸出相對應的 C++原始碼。如此一來，即使沒有專業程式語言背景的使用者，亦可輕鬆利用本系統快速地開發應用程式或建立其雛型。最後，我們以球體之建立、影像處理與表面呈像等三個實例，來說明本系統之可行性。

關鍵字：視覺化程式設計、VTK、電腦製圖、影像處理。

1. 前言

VTK（Visualization Toolkit）是一個開放原始碼且具有跨平台特性的物件導向軟體系統，它的功能包含**三維電腦製圖**（3D computer graphics）、**影像處理**（image processing）以及**視覺化**（visualization）等[1]。VTK 的系統架構由兩個部份組成，一個是以 C++所開發並經過編譯的系統核心（compiled core），另一個為將系統核心封裝起來的**直譯層**（interpreted layer），讓使用者除了使用 C++之外，也能透過 Tcl、Python 和 Java 等語言以直譯的方式來使用 VTK。系統核心能有效率地執行相關的演算法，而直譯層則是協助使用者快速開發程式並建立應用程式的**雛型**（prototyping）。

使用者可以透過 C++或 Tcl、Python 和 Java 等直譯語言來使用 VTK，但前提是要了解這些語言的語法和如何使用 VTK 函式庫，否則就必須花費許多時間學習。所以使用者要操作 VTK 必須具備物件導向程式設計的能力以及對 3D 電腦製圖、影像處理和視覺化要有基本的認識。

程式開發者可以透過視覺化程式設計或框架的形式來開發 VTK 相關之應用程式，視覺化程式設計的優點是直覺、簡單容易操作，框架的優點是沒有限制且可重複利用現有的架構。開發者可以透過不同的平台來建立視覺化程式設計的環境，也能將其應用在不同領域上，[2]是透過 C 語言當作開發平台，並將視覺化程式設計的方式應用在流程圖上；[3]透過 C++語言開發一個視覺化的工具 ITKBoard，此工具將視覺化程式設計的方式應用在 ITK 處理醫療影像分析上；[4]則是利用 Matlab 平台以視覺化程式設計的方式操作 ITK 與 VTK 的功能；[5][6]利用 Java 語言開發一個視覺化的工具 VisPro；[7]比較四種透過框架方式來利用 ITK 功能的軟體。

本論文提出一個以視覺化方式建立圖形化程式設計環境的系統—VPVTK（Visual Programming based on VTK），該系統可以讓使用者在不需要了解特定程式語言的語法之情況下，透過直覺式圖形化的接線與圖示建構出由 VTK 物件所組成的執行管線，最後能將使用者編輯完成之執行管線編譯執行並產生執

行結果。

開發本系統之目的是希望使用者可以利用視覺化方式快速的設計 VTK 執行管線，並在建立執行管線過程中，由系統提供參數設定對話盒讓使用者可以簡單設定 VTK 物件的參數，當系統啟動執行管線後能將前一個 VTK 物件的輸出端與下一個 VTK 物件的輸入端做簡單的資料型別驗證。如此即可讓使用者不需懂得程式語言的語法，以視覺化的操作方式使用 VTK 功能，並節省使用者學習如何操作 VTK 所耗費的時間。

本論文之其餘部分說明如下：第二節說明相關背景與技術，包括**視覺化程式設計**（visual programming）、VTK 和**使用者介面之互動型態**（interaction styles）；第三節為系統分析與設計，介紹本系統之功能需求和設計；結果與討論則在第四節說明，包含系統開發工具與環境、使用者介面、操作方式和實際操作的參考範例；第五節則為結論與未來展望，說明目前的研究成果以及未來努力方向。

2. 相關背景

2.1. 視覺化程式設計

視覺化程式設計是一種以圖形化的表達方式，來顯示並開發電腦程式[8]。許多視覺化程式設計的開發會利用方塊（代表某個屬性或物件的圖示）與連接線（可以將方塊與方塊之間互相連接的工具）當作開發程式的物件。

開發人員可以透過視覺化的操作方式移動選擇的物件，進行物件上的排列組合，建立出執行管線或結構圖，並將完成結果編譯或啟動達到程式開發的目的。

目前廣泛的被應用於工業自動化之領域上，由美商國家儀器（national instruments）所開發的圖形化程式編譯平台 LabVIEW（Laboratory Virtual Instrumentation Engineering Workbench）[9]就是應用視覺化程式設計，提供圖形化的使用者介面開發、資料擷取、儀器控制、報表製作與檔案 I/O 功能，擁有簡單易懂的開發介面，縮短開發速度，比起傳統非圖形化程式編譯平台更容易學習。

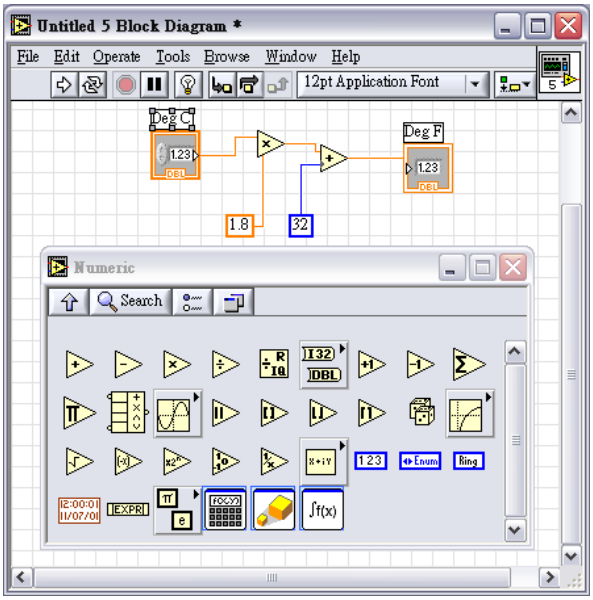

圖 1 LabVIEW 之程式方塊圖提供邏輯運算，將攝氏溫度*1.8 加上 32 最後輸出華氏溫度

2.2. VTK

VTK 是一個開放程式碼且免費的軟體系統，具有跨平台的特性，其功能包括三維電腦製圖、影像處理以及視覺化等[10]。VTK 幾乎可在所有的作業系統環境實作（Unix-base、PC's、Mac OXS Jaguar）[11][12]，其函式庫是由 C++類別函式庫所組成且完全符合物件導向的設計原則。另外，VTK 擁有較高階的抽象化直譯方式，比 OpenGL[13]與 PEX[14]更容易學習與應用。

VTK 建立執行管線的流程是由**視覺化模型**（visualization model）和**圖形化模型**（graphics model）兩個子系統所組成。視覺化模型主要是以**資料流**（data flow）的方式，將資料（例如影像資料）轉換成**圖形資料**（graphical data）。在這個模型中有兩種基本類型的物件，**資料物件**（data object）與**處理物件**（process object）。圖形化模型主要是將視覺化模型輸出的圖形資料轉換成可以顯示的圖片或影像。其主要是由**場景**（scene）、場景中的物體、場景中的光線等核心物件所組成，將上述這些物件組合在一起，便可產生繪圖的場景與視窗，將最後的影像顯示出來。

VTK 的執行管線是由視覺化模型和圖形化模型組成，每一個物件的輸出資料作為下一個物件的輸入，如圖 2 所示。

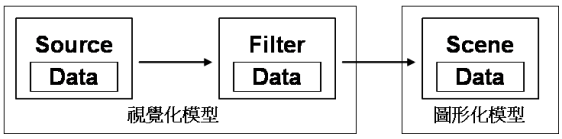

圖 2 VTK 執行管線

2.3. 使用者介面之互動型態

使用者介面之互動型態是包含使用者介面的操作外觀和相關狀態 [15]，使用者與電腦系統之間透過鍵盤或滑鼠的輸入以及螢幕或喇叭的輸出互動。常見的使用者介面互動型態包含下列五種：

一、**命令列**（command line）介面讓使用者以文字的型態利用鍵盤直接輸入指令，是最早使用的互動型態。

二、**功能表選擇**（menu selection）是讓使用者在圖形化的操作介面上利用滑鼠點選方式，讓使用者選擇需要的選項。

三、**表單填寫**（form-fill）是利用表格填寫的方式讓使用者輸入資料。

四、**直接操作**（direct manipulation）允許使用者以直覺的方式與操作介面互動，包含滑鼠的**拖放**（drag and drop）。拖放的概念是指使用者利用滑鼠按鍵選擇指定的來源虛擬物件，將其拖至不同位置或不同虛擬物件的動作，進行指定物件的移動、複製和貼上等操作。

五、**擬人化**（anthropomorphic）為自然語言的介面，此互動形態與人類和其他人互動的方式相同，例如：透過手勢、臉部表情或眼睛移動與電腦系統互動。

3. 系統分析與設計

3.1. 系統分析

圖 3 為本系統之使用案例圖，茲說明如下：

一、物件之管理：

本系統以樹狀結構的方式提供不同選項的 VTK 物件來讓使用者選擇，並依照其功能將之分類為**來源**（source）、**處理**（filter）和**場景**（scene）。來源物件是藉由讀取檔案或經由演算法或數學運算所產生一個以上的資料集合之物件，屬於執行管線的起點。處理物件是以一個以上的資料集合為輸入，經過演算法（例如影像處理）的處理，進而產生新的資料集合。場景物件是負責建立場景、輸出（顯示）或寫入執行管線最後的結果，一般為執行管線的終點。

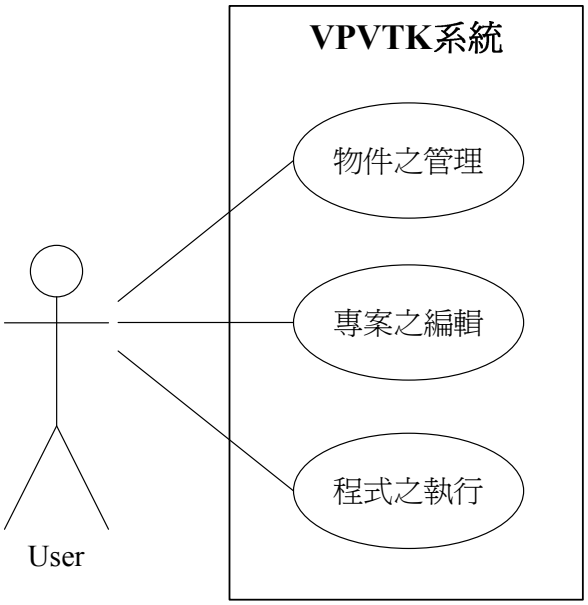

圖 3 使用案例圖

二、專案之編輯：

專案中可能包含一到多條的執行管線，使用者在編輯執行管線的過程中，擁有新增、刪除、排列、連接 VTK 物件與儲存專案的功能。本論文透過類似 Microsoft Office Excel 2003 中工作表的形式讓使用者設計執行管線，是讓使用者能完成執行管線的編輯，而不是著重設計如何排列 VTK 物件以及物件的連接；而在工作表當中預留空間，是為了方便 VTK 物件的連接。VTK 執行管線之執行順序是由左至右，上下則無限制執行順序。

專案編輯過程中，使用者能選擇指定**欄位**（cell）來增加、刪除表單中的行或列，也能設定欄位相對應之參數。使用者能透過系統提供的參數設定對話盒來檢視或設定相對應物件的屬性。由於 VTK 類別參數繁多，本系統會提供部分參數之預設值，以節省使用者編輯專案的時間。

三、程式之執行：

當使用者啟動建立好之執行管線時，系統會開始進行資料型別的驗證，判斷執行管線中前一個 VTK 物件輸出端的資料型態是否與下一個 VTK 物件輸入端能接受的資料型態相符。若驗證的結果皆相符，系統會將驗證成功之訊息輸出，並顯示或產生最後的執行結果（顯示影像或產生 C++原始碼），否則會輸出驗證錯誤的訊息。

3.2. 系統設計

3.2.1. 使用者介面

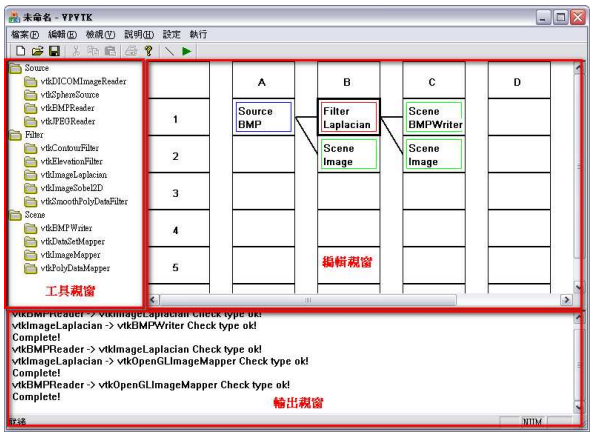

圖 4 本系統之操作介面圖，左上為工具視窗，右上為編輯視窗，下方為輸出視窗

圖 4 為本系統之操作介面，其中包含了三個視窗，工具視窗、編輯視窗和輸出視窗。工具視窗主要是讓使用者以拖放的方式選擇 VTK 物件，編輯視窗主要是將工具視窗提供的 VTK 類別利用圖形化的方式做編輯及排列組合，不同物件擁有各自代表的圖形，以顯示不同顏色不同名稱做為物件的區別。輸出視窗主要是負責將程式執行的狀態訊息輸出，輸出的內容為資料型別驗證的訊息和執行管線是否執行成功的訊息。

圖 5 圖 6 為本系統之參數設定對話盒，使用者可以透過滑鼠右鍵點選執行管線中 VTK 物件的屬性來設定相對應之參數。

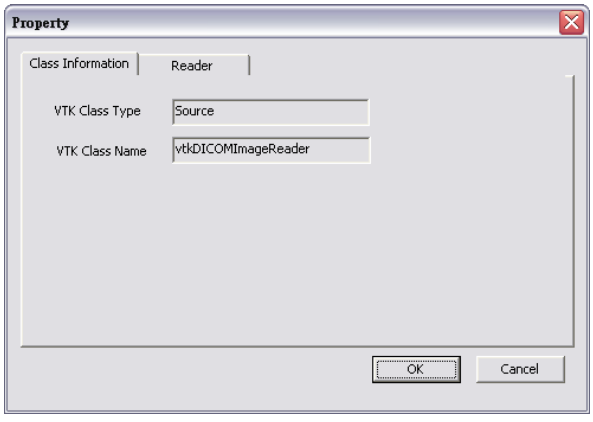

圖 5 物件資訊對話盒，顯示相對應 VTK 物件之型態與名稱

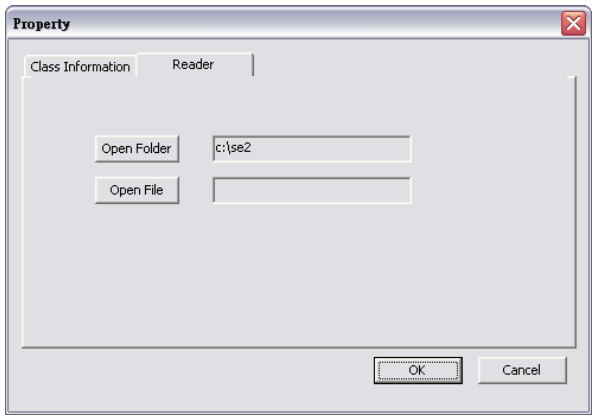

圖 6 讀取路徑對話盒，設定讀取資料夾或檔案之路徑

3.2.2. 類別圖

圖 7 為本系統之類別圖，並說明如下：

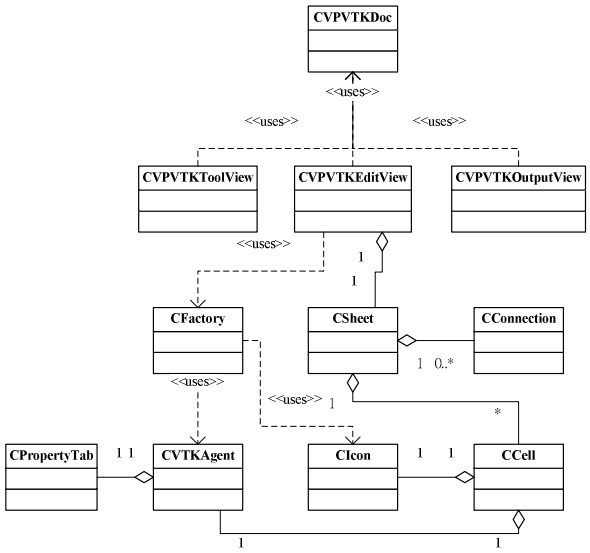

圖 7 類別圖

一、本系統之視窗組成

CVPVTKEditView、CVPVTKToolView 和 CVPVTKOutputView 分別代表系統中的三個視窗。CVPVTKEditView 主要是負責管理編輯視窗的操作，其中包含 VTK 物件在編輯視窗中的操作（移動、複製和刪除）、連接線的管理（建立、更新和刪除）、VTK 執行管線的執行、記錄滑鼠在編輯視窗的狀態（左鍵點下與放開的座標位置）、管理整個表單的顯示（表單的放大或縮小）、增加表單的行或列以及計算某一點在顯示比例為 100%時的原始座標位置。CVPVTKToolView 負責 VTK 物件的顯示以及樹狀結構的分類，可以當作 VTK 物件拖

放的來源端。COutputView 負責程式執行狀態訊息的輸出，包含輸出輸入端的型別驗證結果，以及最後程式是否執行成功都會在此輸出狀態訊息。

二、編輯視窗之組成

編輯視窗由一個表單所組成，表單中又包含了連接線和許多的欄位。其中由 CSheet 負責記錄表單中行與列的總數、行與列的寬度與高度、儲存欄位的**向量**（vector）、表單的顯示比例、預留連接的空間的大小。主要功能有設定行與列的寬度與高度、取得目前表單的面積大小、增加或減少行或列的個數、計算畫欄位所需之左上角點的座標位置。CConnection 負責記錄連接線的兩個欄位，功能為設定繪製連接線的兩個欄位以及繪製連接線。CCell 負責記錄表單中的欄位資訊，包括畫筆型態、畫筆顏色、畫筆寬度、取得畫連接線需要的兩個連接點位置、記錄欄位是否能接受拖放的動作、是否能接受繪製連接線，並且在每一個欄位中有屬於自己的圖示。

三、產生相對應之圖示與代理類別

使用者在工具視窗中選擇 VTK 物件將其拖放至編輯視窗時，必須產生與 VTK 物件相對應之圖示與代理類別。其中以 CFactory 負責記錄 VTK 物件名稱，依照取得的 VTK 物件名稱來產生相對應的 CIcon 物件以及 CVTKAgent 物件。CIcon 負責記錄每一個欄位所對應的圖示，包含圖示的名稱、名稱的顏色、名稱字型的大小、背景的顏色、圖示的寬度以及高度，並且提供設定圖示的名稱、設定名稱的顏色、設定名稱字型的大小、設定背景的顏色圖示繪製的函式。CVTKAgent 是做為 VTK 類別的代理類別，每一個 VTK 類別都有屬於自己本身的 CVTKAgent，主要是記錄此類別所屬的 VTK 類別名稱和 VTK 類別型態，可以產生相對應 VTK 類別的**實體**（instance）、提供參數設定對話盒給對應的 VTK 類別、啟動 VTK 執行管線（屬於 scene 才有效）、檢查對應 VTK 類別輸入端可以接受的資料型態。

四、代理類別與參數之設定

由於本論文需要將不同的 VTK 物件連接且讓每一個 VTK 物件彼此能互相溝通建立執行管線，因此透過代理類別的方式包裝 VTK 所提供的類別。代理類別的命名方式依照 VTK 類別的名稱而定，例如 vtkBMPReader 的代理類別稱為 CAgentBMPReader，而這些代理類別都繼承自 CVTKAgent 類別。由於 VTK 類別本身包含了一些參數需要讓使用者設定，因此本系統透過建立參數設定對話盒的方式讓使用者設定執行管線中每一個 VTK 物件的參數，但 VTK 類別的參數不一，本系統僅列出幾項做為代表，其中 CPropertyTab 是負責參數設定的對話盒類別。

3.2.3. 循序圖

新增 VTK 物件之流程敘述如下：

首先判斷拖放視窗的來源若是屬於工具視窗且置放的欄位是可以接受置放動作的話，則將從工具視窗取得的 VTK 物件名稱傳入 CFactory 中的 SetComponentName 函式，依照取得的 VTK 物件名稱呼叫 CreateCIconObject 函式建立相對應的 CIcon 物件，呼叫 CreateCVTKAgent 函式建立相對應的 CVTK Agent 物件，接著以 SelectIcon 與 SelectVTKAgent 的方式到目的的欄位中設定取得的圖示以及代理類別，再將所有圖示設定為未點選，將目前的圖示設定為已點選，最後再進行 CSheet 的重畫，如圖 8。

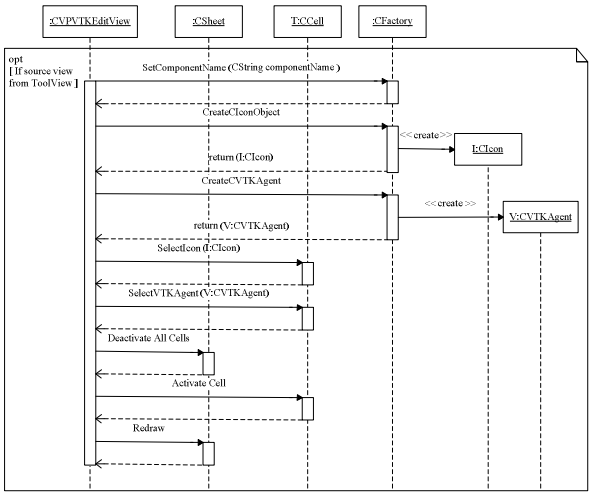

圖 8 新增 VTK 物件之流程圖

3.2.4. 建立執行管線演算法

建立執行管線演算法目的是為了找出編輯視窗中所有的執行管線，並記錄執行管線所屬的欄位，透過欄位即可取得所屬的 VTK 代理類別。**堆疊**（stack）、**雙向串列**（list）和向量都是 STL（Standard Template Library）所提供的容器樣版類別，用來儲存各種標準型別或使用者自定的類別。本系統利用堆疊、雙向串列和向量做為存放資料的基礎，利用堆疊存放我們以縱向優先搜尋演算法找到的欄位，雙向

串列則用來存放找到的執行管線，向量則用來存放所有找到的執行管線。

建立執行管線演算法中，首先會找出編輯視窗左上第一個不為空的欄位並放入堆疊中，判斷若是堆疊不為空，則取出堆疊最上方欄位（在此稱為 pTopCell），利用 pTopCell 找出與其相連的連接線，若有找到連接線則將 pTopCell 設定為已經拜訪過的欄位，將找出之連接線另一端的欄位放入堆疊中，若沒找到連接線則判斷 pTopCell 是否為未拜訪，若是 pTopCell 為之前未拜訪過的欄位，則建立雙向串列 （存放找到的執行管線）與向量 （存放所有找到的執行管線），將堆疊中的欄位由後取出並由前存放到雙向串列中，並將建立好的雙向串列存放至向量中，將 pTopCell 設定為已拜訪過之欄位，最後刪除堆疊最上方的欄位和已拜訪過的連接線，直到堆疊為空演算法才結束。

4. 結果

4.1. 開發工具與環境

本系統之實作環境為 Microsoft Windows XP SP3 平台，以 Microsoft Visual Studio 2005 中的 C++語言進行開發，並搭配 MFC 視窗程式的框架，而 VTK 使用的版本為 VTK 5.4.2。

4.2. 實作結果

本系統之實作結果將介紹三個實作範例，茲說明如下：

表面呈像是三維呈像方式的一種，其主要原理是以多邊形來表示物體的表面特徵，依照物體凹凸的特徵，將表面轉換成由三角形或多邊形所組成的連續面。本範例是將一系列的 DICOM 影像做表面呈像，在建立 VTK 執行管線來源的部份會利用 vtkDICOMImageReader 類別來讀取一系列的 DICOM 影像，處理的部分則是會利用 vtkContourFilter 類別來取得一系列 DICOM 影像的輪廓線，並利用 vtkSmoothPolyDataFilter 類別將影像做平滑處理，場景的部份可選擇 vtkPolyDataMapper 或 vtkDataSetMapper 來建立顯示的場景，實作結果如圖 9。若是以手動輸入程式碼的方式來完成表面呈像，就必須設定使用到之 VTK 標頭檔以及 VTK 函式庫的路徑，如圖 10。

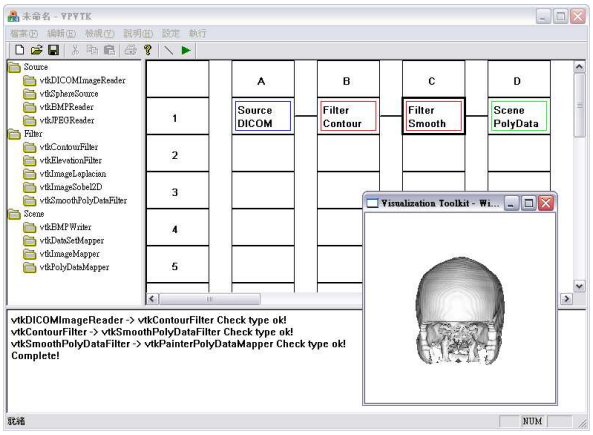

圖 9 表面呈像實作結果

```
#include "vtkRenderer.h"
#include "vtkRenderWindow.h"
#include "vtkRenderWindowInteractor.h"
#include "vtkPolyDataMapper.h"
#include "vtkActor.h"
#include "vtkCamera.h"
#include "vtkContourFilter.h"
#include "vtkDICOMImageReader.h"
#include "vtkSmoothPolyDataFilter.h"

reader = vtkDICOMImageReader::New();
reader->SetDirectoryName("c:\\se2");
bone = vtkContourFilter::New();
bone->SetInput((vtkDataSet*)reader->GetOutput());
bone->SetValue(0,300);
boneSmooth = vtkSmoothPolyDataFilter::New();
boneSmooth->SetInput(bone->GetOutput());
boneSmooth->SetNumberOfIterations(60);
boneSmooth->SetRelaxationFactor(0.05);
boneMapper = vtkPolyDataMapper::New();
boneMapper->SetInput(boneSmooth->GetOutput());
boneMapper->ScalarVisibilityOff();

actor = vtkActor::New();
actor->SetMapper(boneMapper);
aCamera = vtkCamera::New();
aCamera->SetViewUp (0, 0, -1);
aCamera->SetPosition (0, 1, 0);
aCamera->SetFocalPoint (0, 0, 0);
aCamera->Dolly(1.5);
aRenderer = vtkRenderer::New();
aRenderer->AddActor(actor);
aRenderer->SetActiveCamera(aCamera);
aRenderer->ResetCamera ();
aRenderer->SetBackground(1,1,1);
aRenderer->ResetCameraClippingRange();
renWin = vtkRenderWindow::New();
renWin->AddRenderer(aRenderer);
iren = vtkRenderWindowInteractor::New();
iren->SetRenderWindow(renWin);
renWin->SetSize( 300, 300);
renWin->SetParentId(this->GetSafeHwnd());
renWin->Render();

設定 library
"C:\Program Files\Microsoft Visual Studio 8\VC\lib\vtk"
```

圖 10 表面呈像程式碼

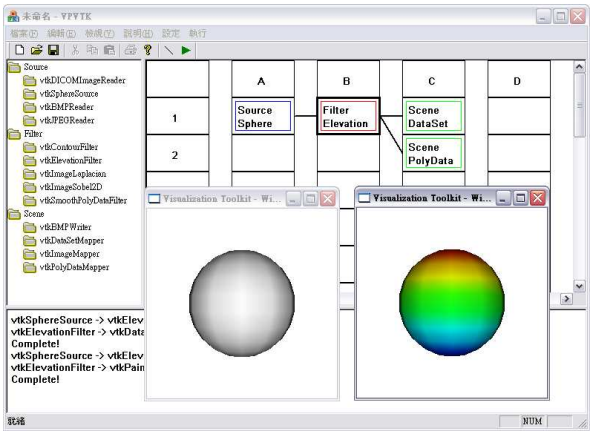

圖 11 球體建立實作結果

圖 11 為球體建立的實作結果，此範例在建立 VTK 執行管線時來源的部份會利用 vtkSphereSource 類別來讀取一個 VTK 所提供的球體，處理部分可以選擇 vtkElevationFilter 類別將讀取的球體做濾波處理，場景的部份我們選擇 vtkPolyDataMapper 和 vtkDataSetMapper 建立出兩個顯示的場景。

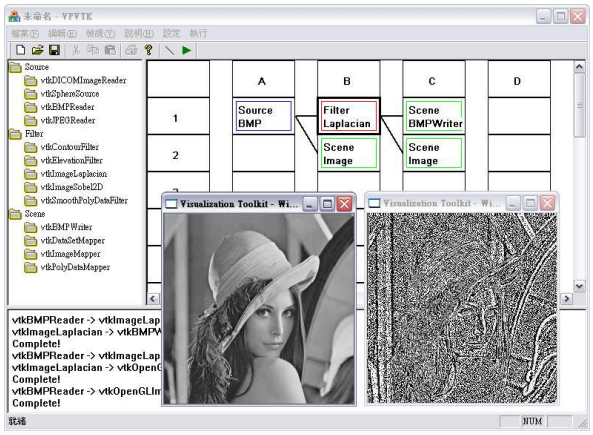

圖 12 影像處理實作結果

圖 12 為影像處理之實作，此範例在建立 VTK 執行管線時來源的部份可以選擇 vtkBMPReader 或 vtkJPEGReader 類別來讀取一個影像檔案，處理部分則是可以選擇 vtkImageSobel2D 或 vtkImageLaplacian 類別將讀取的影像做處理，場景的部份可選擇 vtkImageMapper 類別來顯示影像或 vtkBMPWriter 類別將影像寫入檔案，實作結果如圖 12。

5. 結論

本研究開發一個 VPVTK 的系統，讓使用者以視覺化的方式操作 VTK 功能，也描述開發本系統的設計與分析方法，並且透過三個範例實際操作本系統，下列將敘述本系統之優點。

- 不需懂得程式語言的語法：使用者透過本系統視覺化的操作介面，替代傳統手動輸入程式碼的方式，達到使用 VTK 功能的目的，也節省學習使用 VTK 所耗費的時間。
- 快速的設計執行管線：本系統允許使用者利用視覺化的方式透過圖形化的使用者介面建立由 VTK 物件所組成的執行管線。
- 簡單的參數設定：本系統提供參數設定對話盒讓使用者可以簡單的設定 VTK 物件的參數。
- 簡單的資料型別驗證：本系統啟動執行管線的過程中會將前一個 VTK 物件的輸出端與下一個 VTK 物件的輸入端做簡單的資料型別驗證。
- 具有延展性的：若是系統中要增加新的 VTK 功能，可以透過新增 VTK 代理類別的方式，完成系統功能的擴充。

本系統已能達到透過視覺化方式使用 VTK 功能的目的。未來我們希望提供使用者能將編輯好的執行管線做儲存和讀取，此功能可以讓使用者下次開啟 VPVTK 系統時讀取之前所編輯的執行管線，幫助使用者節省重新建立執行管線的時間。由於 VTK 本身支援之類別過於龐大，目前本系統僅提供幾個 VTK 類別，未來我們希望可以將本系統擴充延展支援更多的 VTK 類別，並且將 ITK(Insight Segmentation and Registration Toolkit）整合至系統中。

參考文獻

[1] William J. Schroeder, Lisa S. Avila and William Hoffman, “Visualizing with VTK: A Tutorial,” ***IEEE Computer Graphics and Applications***, Vol. 20, No. 5, pp. 20-27, 2000.

[2] Kanis Charntaweekhun and Somkiat Wangsiripitak, “Visual Programming using Flowchart,” ***Communications and Information Technologies***, pp. 1062-1065, 2006.

[3] Hoang D. K. Le, Rongxin Li, and Sebastien Ourselin, “Towards a Visual Programming Environment Based on ITK for Medical Image Analysis,” ***Digital Image Computing: Techniques and Applications***, pp. 558- 565, 2005.

[4] Gobbi D., Mousavi P., Li K., Xiang J., Campigotto A., LaPointe A., Fichtinger G. and Abolmaesumi P., “Simulink Libraries for Visual Programming of VTK and ITK,” ***Systems and Architectures for Computer Assisted Interventions***, 2008.

[5] Da-Qian Zhang and Kang Zhang, “VisPro: a visual language generation toolset,” ***IEEE Symposium on Visual Languages***, pp. 195-202, 1998.

[6] Kang Zhang, Da-Qian Zhang and Jiannong Cao, “Design, Construction, and Application of a Generic Visual Language Generation Environment,” ***IEEE Transactions on Software Engineering***, Vol. 27, No. 4, pp. 289-307, 2001.

[7] Ingmar Bitter, Robert Van Uitert, Ivo Wolf, Luis Ibanez and Jan-Martin Kuhnigk, “Comparison of Four Freely Available Frameworks for Image Processing and Visualization That Use ITK,” ***IEEE Transactions on Visualization and Computer Graphics***, Vol. 13, No. 3, pp. 483-493, 2007.

[8] Margaret M. Burnett, Adele Goldberg, Ted G Lewis, ***Visual object-oriented programming***,

Manning Publication Co, 1995.

[9] LabVIEW, http://www.ni.com/labview/zht/, 2009.

[10] VTK Homepage, http://www.vtk.org/, 2009.

[11] Lisa S. Avila, Sebastien Barre, Rusty Blue, Berk Geveci, Amy Henderson, William A. Hoffman, Brad King, C. Charles Law, Kenneth M. Martin, William J. Schroeder, ***The VTK User's Guide***, Kitware, 2004.

[12] Will Schroeder, Ken Martin, Bill Lorensen, ***The Visualization Toolkit - An Object-Oriented Approach to 3D Graphics***, Kitware, 2003.

[13] OpenGL, http://www.opengl.org/, 2009.

[14] PEX, http://research.microsoft.com/en-us/projects/pex/, 2009.

[15] Debbie Stone, Caroline Jarrett, Mark Woodroffe, Shailey Minocha, ***User Interface Design and Evaluation***, Morgan Kaufmann, 2005.