**計算機圖學單元介紹**

1. 英文主題：

Chapter 01: Open a window

1. 中文主題：

單元01：創建視窗

1. 組別：

第00組 (老師)

1. 組員：

0000\_魏志達

1. 作業分工：

(詳見作業報告) …略

1. 功能簡述：

本單元介紹一系列最基本的API函數，讓OpenGL建立並開啟繪圖視窗。

本課程所使用的接口程式庫為'OpenGL utility library' 簡稱GLUT。它制定了一套可供高階語言呼叫執行的API，藉此達到視窗管理、事件處理、IO 控制等功能。這些函數已經類似於函式庫(Lib)的存在，可與主程式共同編譯為二進位碼，但它仍是與標準化的硬體介面互動的有效工具，所以習慣上仍稱之為API函數。本課程所使用的GLUT，是一套跨平台的程式庫，這讓程式設計師可以在支援GLUT的不同硬體、不同作業系統、以及不同語言之間自由移植。除了GLUT之外，其他接口程式庫包括SDL 和GLFW等，可部份取代GLUT，或與GLUT共同使用。

1. 主要程式碼：

本單元為第一次程式實作，為完整解釋程式流程，故將全部原始碼詳列如下，

相關檔案：Ch\_01\_tm0\_src1.cpp

|  |
| --- |
| #include <GL/freeglut.h>  static void RenderSceneCB()  {  glClear(GL\_COLOR\_BUFFER\_BIT);  // do something here  glutSwapBuffers();  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGBA);  glutInitWindowSize(768, 512);  glutInitWindowPosition(300, 200);  glutCreateWindow("Tutorial 01 A");  glutDisplayFunc(RenderSceneCB);  glClearColor(0.0f, 0.0f, 0.0f, 0.0f);  glutMainLoop();    return 0;  } |

1. 程式說明：
2. 執行OpenGL，在標頭檔部份，只需引用<GL/freeglut.h>即可，往後如需使其他數學與時間函數，可自行引入，但皆與OpenGL繪圖無關。
3. 我們直接由 main() 主程式開始跑流程；
4. glutInit(&argc, argv); 用來初始化GLUT，這裡它的參數直接由main()抄過來，main()執行時如果下達 '-sync' 或 '-gldebug' 等參數，就可以直接帶入這裡進行初始化，而達到監視與除錯等目的，這些專門的參數可以自行在網路上查詢。

註：執行glutInit()之後，就會在記憶體上設置一段GLUT核心區域，記錄供各種GLUT API之間互通訊息的狀態(state)值與指標(index)值，state參數設定值包括目前的緩衝模式、色彩定義、畫筆顏色等等，index則可指向另外產生的畫布區段、程式碼等等。所有GLUT的工作方式皆以「設定—動作—設定—動作—」的流程在運作，例如設定顏色之後，再去繪製指定的點或線或多邊形，而不是同時告訴它用什麼顏色去繪製什麼東西，如果你嫌這樣麻煩，可自己用高階語言的語法將一連串動作與構想形態包裝成可重複使用的物件(object)。注意，對於OpenGL和GLUT而言，只有幾何物件(Geometric object)，如點、線、多邊形等，而沒有程式物件這種相對複雜的概念，這讓跨平台與跨世代的實現更加容易而且穩定。

1. glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA); 用來設定GLUT核心區域中的兩項狀態參數，再次強調，我們不是在write/compile的階段進行這些宣告並編譯為程式碼，而是在執行時，才以「設定—動作—設定—動作—」的方式運作。GLUT\_DOUBLE和GLUT\_RGBA都是在<GL/freeglut.h>已定義的公用常數，但在執行時才設定給GLUT知道。GLUT\_DOUBLE 是指設置雙緩衝（double buffering，即當一個buffer 顯示的時候，另一個buffer 用來繪製）而GLUT\_RGBA 是指色彩模式是RGBA，多了一個透明度(A)，做此設定之後，如果用到色彩，就要用四維描述，比GLUT\_RGB多了一維。
2. 接下來還有2個狀態參數值要設定，分別是視窗大小(w,h)，以及出現的位置，

glutInitWindowSize(768, 512);

glutInitWindowPosition(300, 200);

1. 以上4個狀態值設完成後，接著執行 glutCreateWindow("Tutorial 01 A"); 這仍是一個GLUT的設定API (或是廣義的動作API)，因為它仍只是在幕後工作，並沒有實際繪圖效果；其功能是依據上面狀態值的定義，建立一個繪圖視窗，或者說，是向系統取得(allocate)一塊記憶體區段(或稱緩衝區、畫布)，並在GLUT核心區域記錄一個index，完成畫布的註記。
2. glutDisplayFunc(RenderSceneCB);

這個API是把一段我們自已撰寫的程式 'RenderSceneCB()' 其程式碼的指標值(index)寫到GLUT核心區域的「待執行程序」位置。函數內傳函數指標，是一項高級的程式撰寫技巧，它在其他方面有各種變化與運用方式，而在OpenGL的常用技法中，我們藉此在GLUT的紀錄區域寫上我們所開發的程式碼，以待後續呼叫執行。

1. glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

glutMainLoop();

最後這兩段程式正式開始繪圖，因為執行前會先清除畫布上的所有畫素，glClearColor(0.0f, 0.0f, 0.0f, 0.0f) 這個GLUT的設定API，此時用來設定顏色的四個通道值(RGBA)，必須介於0.0–1.0 之間；而glutMainLoop() 則是一個GLUT的動作API，它將控制權移送給GLUT，並開始它自己的內部循環。此前，我們曾用glutDisplayFunc(RenderSceneCB)在GLUT的核心區域設定一段我們自己撰寫的程式 'RenderSceneCB()' 作為它的「待執行程序」，現在GLUT取得控制權之後就會去執行這段程式。

註：你可以試試改用glClearColor(0.9f, 0.9f, 0.7f, 0.0f) 設定清除畫布所用的顏色(圖一)。

glutMainLoop() 現在做三件事：1.實際顯示出glutCreateWindow("Tutorial 01 A")所建立的視窗，2.以glClearColor(0.0f, 0.0f, 0.0f, 0.0f)的設定清除視窗所對應的記憶體內容(或稱緩衝區、畫布)，3.執行之前指定的自撰程式 'RenderSceneCB()'，它的內容如以下(9)所述。

1. static void RenderSceneCB(){

glClear(GL\_COLOR\_BUFFER\_BIT);

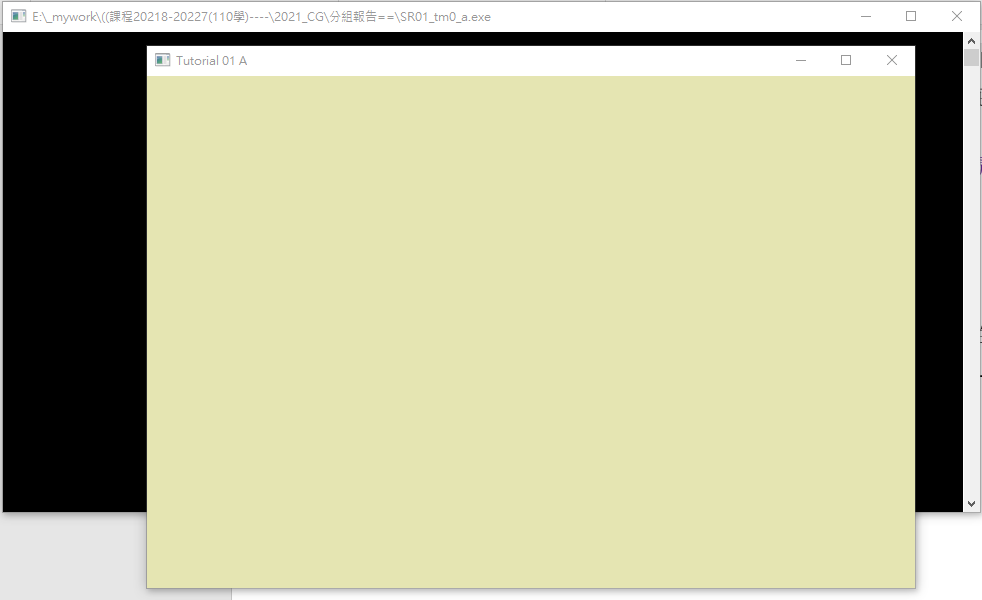
// do something here

glutSwapBuffers(); }

這段我們要交付給GLUT執行的主要程式碼，是一段典型的繪圖程式，還記得此前glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA)的宣告嗎？

GLUT\_DOUBLE是雙畫布緩衝區的意思，亦即，你手上永遠保留舊作，你在上面修改它，隨時畫好，就以glutSwapBuffers() 把它丟出去給GLUT顯示，隨畫隨丟，畫好畫滿，直到你高興為止。但有時候，我們不想在舊作上面繼續畫，而是想在畫完一張之後馬上清除重畫，這樣工作反而單純，不必擦除留在舊畫上的內容，此時，會用glClear(GL\_COLOR\_BUFFER\_BIT)進行清除。注意，這兩行函數都是GLUT的動作API，兩者中間是我們要寫的繪圖程式，但目前沒寫任何內容，所以會開出一個只有背景顏色的空視窗。

註：繪圖時加上內容往往比擦去容易，所以重畫是目前的主流做法，上述空白的內容，未來可以是其他加點、加線、加多邊形、以及貼圖等動作API，由此構成一幅畫面。GLUT沒有擦去特定內容的API，必要時是以背景在該處重繪，優點是修改內容極快，缺點則是，與其他內容重疊時，很難不影響到畫面的完整性。



圖一、以glClearColor(0.9f, 0.9f, 0.7f, 0.0f) 設定清除畫布所用的顏色。

1. 延伸應用程式碼： (Ch\_01\_tm0\_src2.cpp)

|  |
| --- |
| #include <GL/freeglut.h>  static void RenderSceneCB()  {  GLfloat i; // instead of "float i;" thus we allow the "portability"    glClear(GL\_COLOR\_BUFFER\_BIT);  glutSwapBuffers();    for(i=0;i<=1;i+=0.0002){  glClearColor(i, i, 0.7f, 0.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  glutSwapBuffers();  }    glClearColor(0.7f, 0.7f, 0.7f, 0.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  glutSwapBuffers();  }  static void InitializeGlutCallbacks()  {  glutDisplayFunc(RenderSceneCB);  }  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGBA);  glutInitWindowSize(512, 384);  glutInitWindowPosition(200, 200);  glutCreateWindow("Tutorial 01 B");  InitializeGlutCallbacks();  glClearColor(0.9f, 0.7f, 0.9f, 0.5f);  glutMainLoop();    return 0;  } |

1. 應用說明：

主要變動有三處

1. 新增一個中介函數，讓main()先啟動下式之後，再在裡面呼叫glutDisplayFunc();

static void InitializeGlutCallbacks()

{ glutDisplayFunc(RenderSceneCB); }

這是Etay Meiri 的寫作風格，在此還原他的做去，以後看到同學這樣用時，才能適應。

1. 在自行撰寫的主要繪圖程式碼 ’RenderSceneCB()’ 中，雖然還是沒畫什麼東西，但我們執行一個for迴圈，不斷變更顏色設定值，再重新刷新畫布，可以看到視窗背景顏色不斷改變的效果。
2. 在這個迴圈中，變數i的宣告採用GLfloat i; 而非一般C/C++的資料型態 float i;

連接OpenGL時，與繪圖相關的變數，盡量用OpenGL統一定義的型態，以保障可攜性。

十一、其他範例程式：

（無）

十二、注意事項：

自行撰寫的主要繪圖程式碼 ’RenderSceneCB()’，先藉由glutDisplayFunc(RenderSceneCB)將其編譯後所在的index註記到GLUT核心區域，作為預備執行的程序，而後，當main()主程式執行glutMainLoop() 時，GLUT獲得控制權，便會去執行這段繪圖程式碼。

因此 ’RenderSceneCB()’ —或由程式設計師自行命名—非常重要，而且可以變化出更複雜的功能，例如，在AutoCAD或者電影CGI(Computer Generated Imagery)編緝軟體中，可加上使用者輸入、拉線、調整等互動功能，其程式開發方法，必須另外設計自己的一套user interface (UI) 介面，藉由Global Variables與RenderSceneCB() 產生資料交流，並由RenderSceneCB()呼叫GLUT的API把效果繪製到畫布上。有趣的是，當我們想調整畫面時，雖然GLUT最基本會告訴你使用者點了這個位置，滑鼠移到何處，但卻是我們的 UI知道使用者到底想做什麼，然後重新計算畫面應該相對產生什麼效果，再去畫出來，GLUT只是被告知該畫什麼，它不會知道、也不會去管使用者正在做什麼動作\*。

(<https://www.youtube.com/watch?v=e6yKWQ24rF8#t=34s>)

我們還會把構圖想法，以及包含物理公式的呈現規則等，包裝成可重複使用的物件(object)，讓使用者可以很方便的添加某些角色內容到畫布上，所以，當你看到CGI電影動畫公司，用獨家開發的軟體在炫技時，通常是東西一塊塊長出來，長出來之後開始做隨風搖擺等動作，而少見到內容逐一被消除的片段(除非回放)，現在你應該知道那背後是什麼回事！

(<https://www.youtube.com/watch?v=e6yKWQ24rF8#t=2m56s>)

\*註：

(1) 所以動畫軟體通常讓你去點選移動錨點，然後再重新產生對應的效果，而不是直接讓你修改細節；

(2) 畫的東西如何呈現透明與反光的效果，以及從不同角度與距離看起來的畫面是什麼，GLUT會負責幫你搞定。

十三、參考資料：

請自行查詢上述GLUT常數的定義，並由查詢結果延伸了解更多GLUT的公用常數。

例如，資料型態定義表，可參考https://www.khronos.org/opengl/wiki/OpenGL\_Type