b01902068
黃柏智

Lab Report

XBOX controller

程式如何執行

step1
首先，請下載XBOX的driver
http://www.microsoft.com/hardware/zh-tw/d/xbox-360-controller-for-windows
step2
接上XBOX搖桿
step3
開啟MobileSim的執行檔
step4
開啟XBOXcontroller的執行檔(注意此執行檔要與AriaDebugVC12.dll在同目錄)

如何modify code

step1
開啟名為SimpleController_2012的VC++專案
step2
在專案中更改屬性，讓專案可以include到XBOXcontroller中的include目錄
step3
在專案中更改屬性，在連結器中新增AriaDebugVC12.dll
step4
接下來就能開始改code了

技術內容

1. XInput

XInput是微軟的一套API，它可以用來讀搖桿的數據，最多可以支援到4支controller。如果要拿到搖桿的資訊，需要使用XInputGetState()這個function，他會把搖桿的讀數記錄到一個叫XInput_State的structure, 它包含了另一個structure，叫XINPUT_GAMEPAD，用來紀錄讀數，另外還有一個變數叫

dwpacketNumber，可以表示當前的讀數是新進來的還是舊的。若偵測到dwpacketNumber改變了，表示有新的讀數進來，這時再去XInput_Gamepad中去拿資料並作相對應的事。

在XInput中，有個東西叫做dead zone，把搖桿放開時，由於機械結構的關係，搖桿讀數不可能完全歸零，因此就必須設定dead zone的值，若偵測到的讀數小於dead zone，那搖桿讀數就應該視為0，這樣使用者的體驗才能最佳化。

2. WIN32

我設計了一個視窗，他會顯示搖桿的讀數，並讓使用者可以自行設定某些參數。win32程式的進入點叫做WinMain，他有一個參數叫做HINSTANCE ，是特殊的windows handler，負責儲存主程式父視窗 (parent window) 的 handle。寫win32程式時記得要include Windows.h。現在有幾件事要做：

a. 註冊視窗的class

說明：

第一個參數WNDCLASSEX 包含windows class的資訊，包含了cbSize member, 他定義了structure的大小, 以及hIconSm member, 它包含了一個與window class有關的icon。第三個參數是message的處理函式。第四個參數是hInstance，是本程式的 handle ，或者可以稱作 PID (Process ID)，hInstance 所存儲的數值是唯一的。

```
// Register the window class
HBRUSH hBrush = CreateSolidBrush( backgroundColor );
WNDCLASSEX wc =  //第三個參數為message的dispatch函式
{
    sizeof( WNDCLASSEX ), 0, MsgProc, 0L, 0L, hInstance, nullptr,
    LoadCursor( nullptr, IDC_ARROW ), hBrush,
    nullptr, L"XInputSample", nullptr
};
RegisterClassEx( &wc );
```

b. 開一個windows，作為使用者介面

說明：

CreateWindow()是win32 program的重要函式。有11個參數。

1：class name，也就是名字。

2：window text，作用隨class而異，通常是顯示在螢幕上的文字，通常會變成視窗標題。

3：一些開關型屬性，WS_OVERLAPPED是指有標題和邊框，WS_SYSMENU是標題列有關閉按鈕，WS_VISIBLE當然是指看不看得到了，視窗預設都是不可視的，除非指定WS_VISIBLE或之後呼叫ShowWindow()設定。

4~7：X坐標、Y坐標、寬、高。
8：父視窗，這裡是頂層視窗，沒有父視窗所以填NULL。
9：目錄handle，這個視窗沒有目錄所以填NULL(其實我也沒用過這東西)。
10：hInstance，這是為了相容95、98、ME的參數，Windows NT、2000、XP可填NULL。
11：額外資料，這裡沒有額外資料所以填NULL。

```
// Create the application's window
g_hWnd = CreateWindow( L"XInputSample", L"Controller Dectector",
                       WS_OVERLAPPED | WS_VISIBLE | WS_CAPTION | WS_SYSMENU | WS_MINIMIZEBOX,
                       CW_USEDEFAULT, CW_USEDEFAULT, 750, 650,
                       nullptr, nullptr, hInstance, nullptr );
```

c. 處理message loop
說明：
一般訊息處理有兩種類型，PeekMessage與GetMessage。
PeekMessage是如果沒訊息就False，有訊息就True
GetMessage是有訊息且不為 WM_QUIT 則True，有訊息且為WM_QUIT則False
在這裡用的是PeekMessage

```
// Enter the message loop
bool bGotMsg;
MSG msg;
msg.message = WM_NULL;

while( WM_QUIT != msg.message )
{
    // Use PeekMessage() so we can use idle time to render the scene and call pEngine->DoWork()
    bGotMsg = ( PeekMessage( &msg, nullptr, 0U, 0U, PM_REMOVE ) != 0 );

    if( bGotMsg )
    {
        // Translate and dispatch the message
        TranslateMessage( &msg );
        DispatchMessage( &msg );
    }
    else
    {
        UpdateControllerState();
        RenderFrame();
    }
}
```

在這裡介紹一下微軟內部採用的一套命名法則，叫做匈牙利命名法(Hungarian Notations)。它把型態的簡稱加在變數名稱之前。比方說：假定 value 是一個整數變數，則取名成 iValue; name 是一個字串變數，則取名成 strName、…、等等。微軟公司認為這套準則有益於大型軟體計畫的維護，所以在其 Windows API 中都採用這套命名法。
此外，在win32 program中，若要debug，可以打下面這行
#pragma comment(linker, "/subsystem:console /entry:WinMainCRTStartup")
它可以在視窗後面多顯示一個 console， 方便除錯

3. ARIA

ARIA是mobile robot公司寫的一套API，他可以連上這家公司出廠的機器人，如果沒有接上robot，它會連上機器人的模擬器。ARIA程式的進入點與c/c++相同，都是main。在ARIA程式的開頭，會先開啟robot上的感測儀器，並且新增一個command line argument parser。在這裡遇到一個難題，由於我要把ARIA merge進win32 program，但win32沒有main，因此要拿到argc與argv便需要別的方法。圖中的code可以達成此目的。

```
//get command line argument + 轉型成char**
int argc;
LPWSTR *tmpargv = CommandLineToArgvW(GetCommandLineW(), &argc);
if (NULL == tmpargv) {
    wprintf(L"CommandLineToArgvW failed\n");
}
char **argv = (char**)malloc(sizeof(char*) * argc);
for (int i = 0; i < argc; i++) {
    argv[i] = (char*)malloc(sizeof(char) * 500);
    wcstombs(argv[i], tmpargv[i], 500);
}
```

環境設定好了，現在要來控制機器人的動作。在這邊需要一些ARIA的control function。主要有下列幾個：

(a) robot.setVel2(300, 100); => 左輪動300mm, 右輪動100mm
(b) robot.setRotVel(100); => 以100deg/sec轉動
(c) robot.setVel(150); => 以150mm/sec前進(最好先robot.setRotVel(0))
(d) robot.enableMotors(); => 啟動馬達，在行走前最好先call此function
(e) robot.move(1000); => 往前直走1000mm
(f) robot.setDeltaHeading(45); => Change heading by 45 degrees
(g) robot.setHeading(180); => Telling the robot to turn to 180
(h) robot.stop(); => stop the robot
(i) robot.isMoveDone(); => 偵測move()動作完成了沒，可配合sleep使用
(j) robot.isHeadingDone(5) => 偵測setHeading()完成沒，括號內為指定秒數
(k) ArLog::log(ArLog::Normal, "simpleMotionCommands: Pose=(%.2f,%.2f,%.2f), Trans. Vel=%.2f, Rot. Vel=%.2f, Battery=%.2fV", robot.getX(), robot.getY(), robot.getTh(), robot.getVel(), robot.getRotVel(), robot.getBatteryVoltage()); => 拿position資訊
(l) robot.stopRunning(); => stop background thread
(m) robot.waitForRunExit(); => wait thread to end

值得注意的是，當你需要修改robot的動作時，要先robot.lock()，等動作完成後，重輸入robot.unlock()，這樣才不會發生conflict。
接著介紹我控制機器人行走的方式，我會算出搖桿讀數的cosine值，來決定現在機器人要往哪裡走。因為人無法感測到太精細的角度變化，因此我以30度為單位，將360度切成12個方位，再讓機器人往那邊走。

搖桿上的功能

左右trigger: 機器人的轉向
左搖桿: 控制機器人行走
右搖桿: 讓機器人原地轉固定角度
方向鍵: 上下鍵代表前進與後退，左右鍵代表旋轉
start鍵: 讓機器人停止動作