

Object Oriented Programming
Windows Tutorials
Tutorial #6 : Handle Memory Leaks

Shuo-Han Chen (陳碩漢),
shchen@csie.ntut.edu.tw

Hong-Yue Technology Research Building 1223
F 09:10 - 12:00

Game Framework 4.10使用說明

Visual Studio 2017

練習6：處理Memory Leaks

1. 請先完成練習5
2. 這練習讓使用者按下空白鍵後，隨機產生幾顆CBouncingBall 在地圖上。
3. 在CGameMap 加入新的函式及成員，如下圖。

```
class CBouncingBall;
class CGameMap
{
public:
    CGameMap();
    void LoadBitmap();
    void OnShow();                //顯示地圖
    void OnMove();                //播放彈跳球的動畫
    void OnKeyDown(UINT);         //處理按鍵按下後CGameMap的反應
    void RandomBouncingBall();    //隨機彈跳球的個數加入到Map中
    void InitializeBouncingBall(int, int, int); //初始化彈跳球
    ~CGameMap();                  //解構子
protected:
    CMovingBitmap blue, green;    //建立藍色地圖和綠色地圖
    int map[4][5];                //建立一個地圖矩陣的index
    const int X, Y;               //大地圖左上角的X,Y座標
    const int MW, MH;             //每張小地圖的寬高度
    CBouncingBall* bballs;        //CBouncingBall指標
    int random_num;               //隨機個數
};
```

備註:如果CGameMap 的 class 是放於CBouncingBall 之前，請在CGameMap 上方(如圖)加入class CBouncingBall;。

4.為CBouncingBall 加入新的方法，讓它可以設定起始速度，起始座標以及彈跳水平面。

```
class CBouncingBall
{
public:
    CBouncingBall();
    void LoadBitmap();           // 載入圖形
    void OnMove();               // 移動
    void OnShow();               // 將圖形貼到畫面
    void SetFloor(int);          // 設定彈回的水平面
    void SetXY(int,int);         // 設定起始上升座標
    void SetVelocity(int);       // 設定上升的起始速度
private:
    int x, y;                    // 圖形座標
    int floor;                   // 地板的Y座標
    bool rising;                 // true表上升、false表下降
    int initial_velocity;        // 初始速度
    int velocity;                // 目前的速度(點/次)
    CAnimation animation;        // 利用動畫作圖形
};
```

mygame.cpp 實作部分(CBouncingBall):

```
void CBouncingBall::SetXY(int x,int y)
{
    this->x = x;
    this->y = y;
}

void CBouncingBall::SetFloor(int floor)
{
    this->floor = floor;
}

void CBouncingBall::SetVelocity(int velocity)
{
    this->velocity = velocity;
    this->initial_velocity = velocity;
}
```

mygame.cpp 實作部分(CGameMap)

```
void CGameMap::InitializeBouncingBall(int ini_index, int row, int col)
{
    const int VELOCITY = 10;           //球的起始上升速度
    const int BALL_PIC_HEIGHT = 15;    //球圖片的高度
    int floor = Y+(row+1)*MH-BALL_PIC_HEIGHT; //設定球的落下點為Map的下方

    bballs[ini_index].LoadBitmap();     //載入彈跳球的動畫
    bballs[ini_index].SetFloor(floor);   //設定彈跳的起始水平面
    bballs[ini_index].SetVelocity(VELOCITY+col); //設定彈跳的初始速度，越右邊的彈得越高
    bballs[ini_index].SetXY(X+col*MW+MW/2, floor); //設定球的起始位置X座標為該Map一半的位置
}
```

```
void CGameMap::RandomBouncingBall()
{
    const int MAX_RAND_NUM = 10;        //隨機的最大值
    random_num = (rand() % MAX_RAND_NUM)+1; //隨機1~MAX_RAND_NUM

    bballs = new CBouncingBall[random_num]; //根據隨機值動態配置CBouncingBall陣列
    int ini_index = 0;                      //初始化陣列索引
    for(int row=0;row<4;row++)
        for(int col=0;col<5;col++)
        {
            if(map[row][col] != 0 && ini_index < random_num) //只放球在有色的地圖且初始化的陣列索引必小於隨機的個數
            {
                InitializeBouncingBall(ini_index, row, col);
                ini_index++; //初始化的陣列索引加一
            }
        }
}
```

mygame.cpp 實作部分(CGameMap)

```
void CGameMap::OnKeyDown(UINT nChar)
{
    const int KEY_SPACE = 0x20;
    if(nChar == KEY_SPACE)
        RandomBouncingBall(); //當空白鍵按下後隨機彈跳球
}

void CGameMap::OnMove()
{
    for(int i=0;i<random_num;i++)
    {
        bballs[i].OnMove();
    }
}

CGameMap::~CGameMap()
{
}
```

5. 在mygame.cpp 檔案中的，

(a) CGameStateRun::OnMove

```
void CGameStateRun::OnMove()
{
    .....
    gamemap.OnMove();
    .....
}
```

(b) CGameStateRun::OnKeyDown 加入

```
void CGameStateRun::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    .....

    gamemap.OnKeyDown(nChar);
}
```

(c) CGameMap::OnShow 最下方加入

```
void CGameMap::OnShow()
{
    .....

    for(int i=0;i<random_num;i++)
    {
        bballs[i].OnShow();
    }
}
```

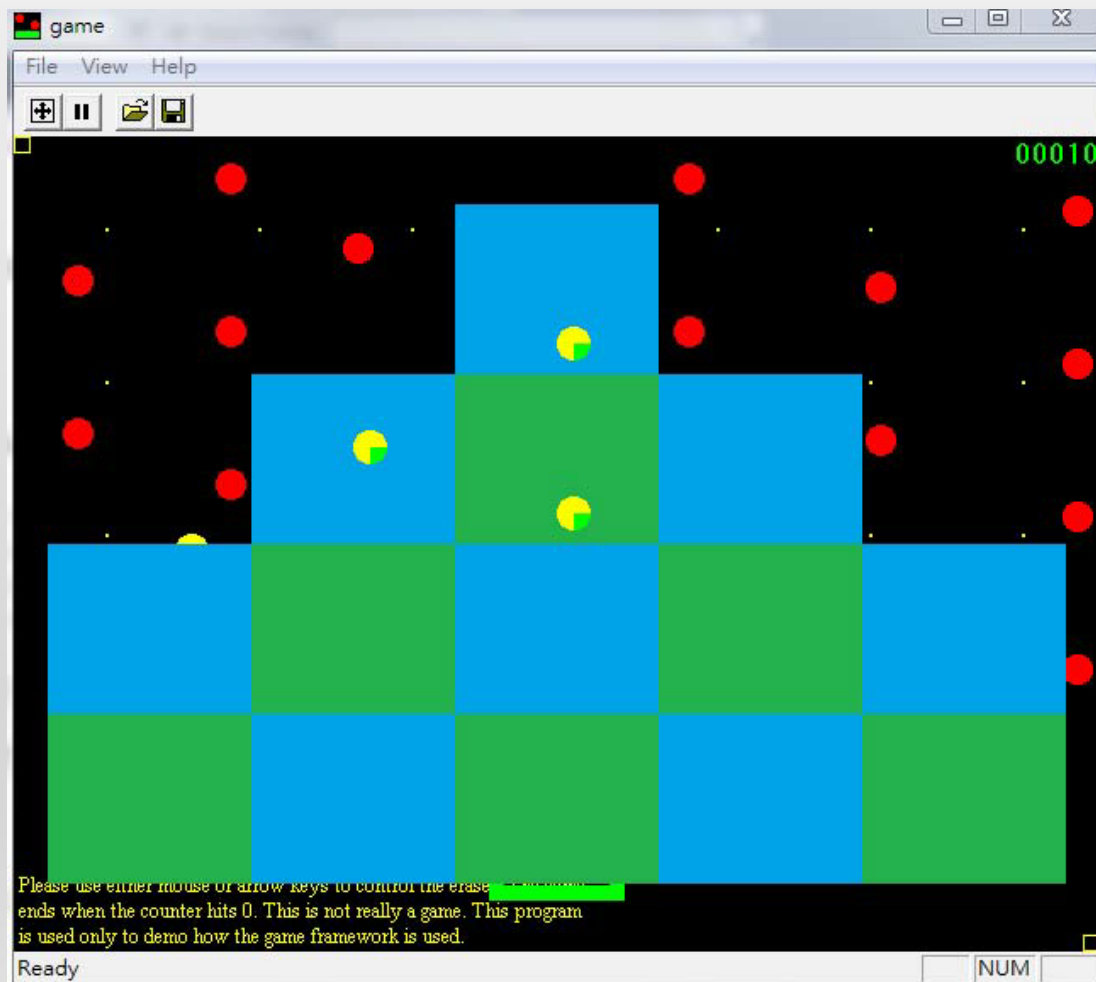
(d) CGameMap::CGameMap 加入

```
CGameMap::CGameMap()
: X(20), Y(40), MW(120), MH(100)
{
    .....

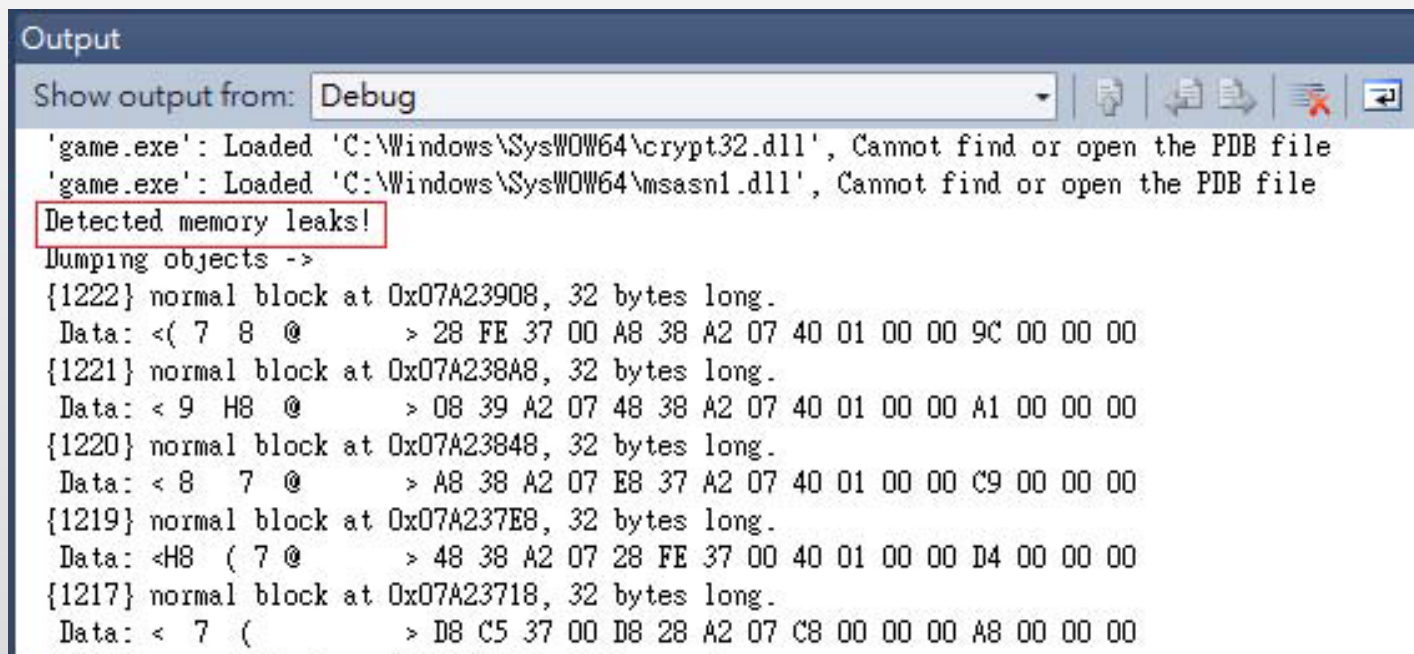
    random_num = 0;    //初始化隨機個數
```


6.檢查Memory leaks

(a) Compile->Run。按下空白鍵後，會有隨機的幾顆球在Map 裡彈跳。



(b)此時關掉game 會發現VS 的 output 視窗裡跑出如圖(memory leaks)。



The screenshot shows the Visual Studio Output window with the 'Debug' filter selected. The output text indicates that 'game.exe' loaded 'crypt32.dll' and 'msasn1.dll' but could not find or open their PDB files. A red box highlights the message 'Detected memory leaks!'. Below this, the text 'Dumping objects ->' is followed by a list of memory blocks, each 32 bytes long, with their addresses and hex data. The blocks are numbered 1222, 1221, 1220, 1219, and 1217 in descending order.

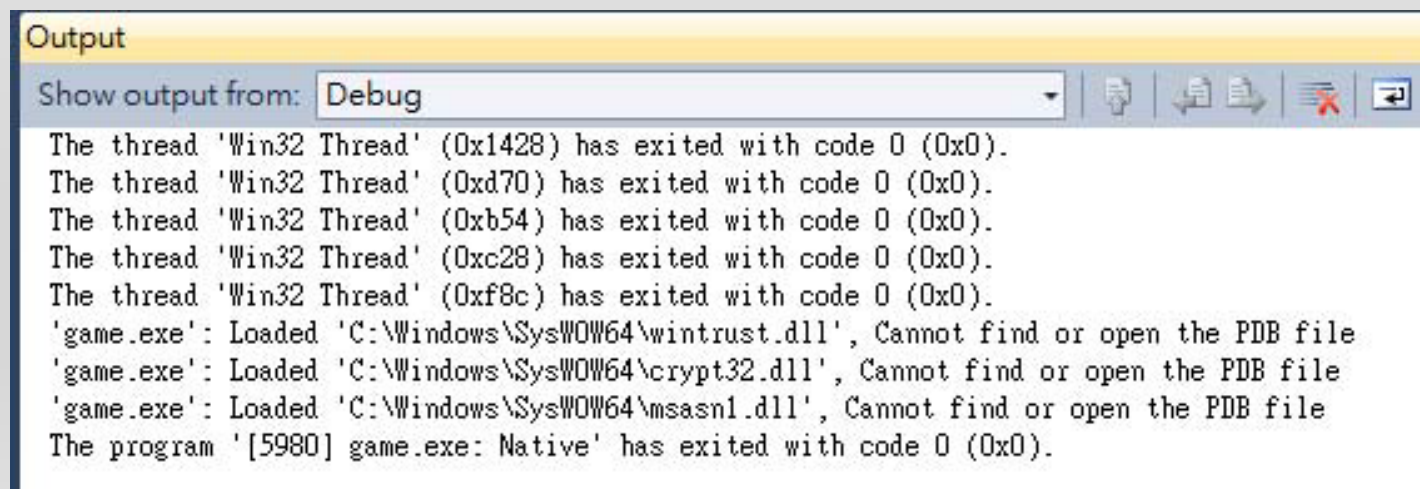
```
Output
Show output from: Debug
'game.exe': Loaded 'C:\Windows\SysWOW64\crypt32.dll', Cannot find or open the PDB file
'game.exe': Loaded 'C:\Windows\SysWOW64\msasn1.dll', Cannot find or open the PDB file
Detected memory leaks!
Dumping objects ->
{1222} normal block at 0x07A23908, 32 bytes long.
Data: <( 7 8 @      > 28 FE 37 00 A8 38 A2 07 40 01 00 00 9C 00 00 00
{1221} normal block at 0x07A238A8, 32 bytes long.
Data: < 9 H8 @      > 08 39 A2 07 48 38 A2 07 40 01 00 00 A1 00 00 00
{1220} normal block at 0x07A23848, 32 bytes long.
Data: < 8 7 @      > A8 38 A2 07 E8 37 A2 07 40 01 00 00 C9 00 00 00
{1219} normal block at 0x07A237E8, 32 bytes long.
Data: <H8 ( 7 @      > 48 38 A2 07 28 FE 37 00 40 01 00 00 D4 00 00 00
{1217} normal block at 0x07A23718, 32 bytes long.
Data: < 7 (      > D8 C5 37 00 D8 28 A2 07 C8 00 00 00 A8 00 00 00
```

7.修正Memory leaks 之一

(a)出現Memory leak 的原因是程式對bballs 作了new，但是卻忘了作delete，因此，應該在解構子釋放動態配置的bballs

```
CGameMap::~CGameMap()  
{  
    delete [] bballs;  
}
```

(b) Compile->Run。按下空白鍵一次後關掉game，memory leaks 訊息消失了

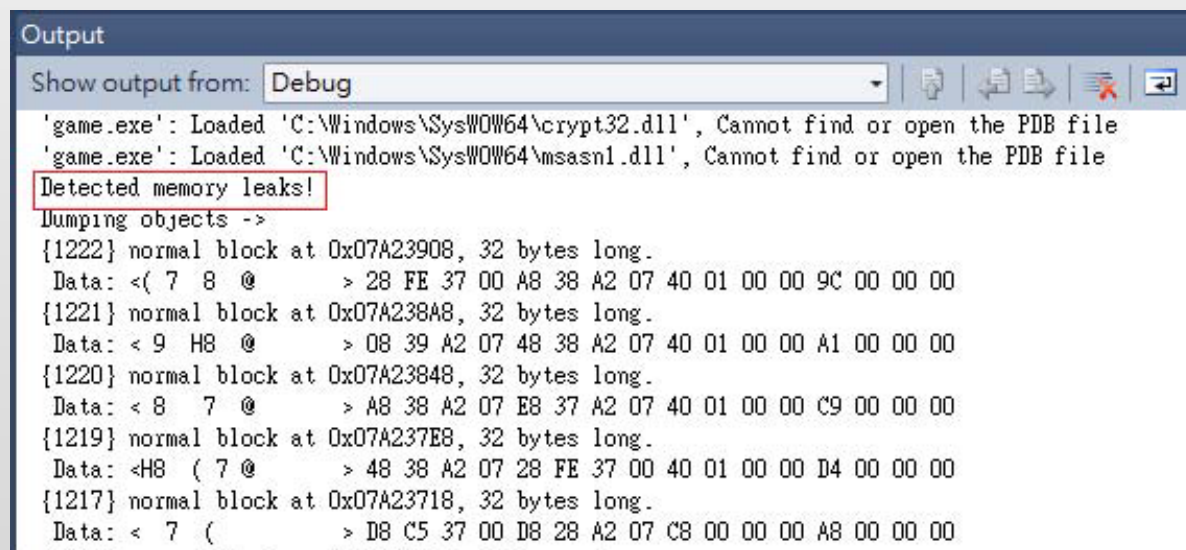


The screenshot shows the 'Output' window in Visual Studio. The 'Show output from:' dropdown is set to 'Debug'. The output text is as follows:

```
Output  
Show output from: Debug  
The thread 'Win32 Thread' (0x1428) has exited with code 0 (0x0).  
The thread 'Win32 Thread' (0xd70) has exited with code 0 (0x0).  
The thread 'Win32 Thread' (0xb54) has exited with code 0 (0x0).  
The thread 'Win32 Thread' (0xc28) has exited with code 0 (0x0).  
The thread 'Win32 Thread' (0xf8c) has exited with code 0 (0x0).  
'game.exe': Loaded 'C:\Windows\SysWOW64\wintrust.dll', Cannot find or open the PDB file  
'game.exe': Loaded 'C:\Windows\SysWOW64\crypt32.dll', Cannot find or open the PDB file  
'game.exe': Loaded 'C:\Windows\SysWOW64\msasn1.dll', Cannot find or open the PDB file  
The program '[5980] game.exe: Native' has exited with code 0 (0x0).
```

7.修正Memory leaks 之二

(a) Compile->Run。按下空白鍵一次，隨機幾顆球後，再按下第二次，又隨機出現幾顆球，這時關掉game，情況如下圖，又出現memory leaks 了。



```
Output
Show output from: Debug
'game.exe': Loaded 'C:\Windows\SysWOW64\crypt32.dll', Cannot find or open the PDB file
'game.exe': Loaded 'C:\Windows\SysWOW64\msasn1.dll', Cannot find or open the PDB file
Detected memory leaks!
Dumping objects ->
{1222} normal block at 0x07A23908, 32 bytes long.
Data: <( 7 8 @      > 28 FE 37 00 A8 38 A2 07 40 01 00 00 9C 00 00 00
{1221} normal block at 0x07A238A8, 32 bytes long.
Data: < 9 H8 @      > 08 39 A2 07 48 38 A2 07 40 01 00 00 A1 00 00 00
{1220} normal block at 0x07A23848, 32 bytes long.
Data: < 8 7 @      > A8 38 A2 07 E8 37 A2 07 40 01 00 00 C9 00 00 00
{1219} normal block at 0x07A237E8, 32 bytes long.
Data: <H8 ( 7 @      > 48 38 A2 07 28 FE 37 00 40 01 00 00 D4 00 00 00
{1217} normal block at 0x07A23718, 32 bytes long.
Data: < 7 (      > D8 C5 37 00 D8 28 A2 07 C8 00 00 00 A8 00 00 00
.....
```

(b)出現問題的原因是「當按一次空白鍵然後關閉時，動態配置出來的球會被解構子所釋放，但是，當按二次空白鍵後再關閉時，解構子釋放的是第二次動態配置的空間，而第一次配置的卻空間沒有被釋放」。因此，必須修改程式，在動態配置新的 bballs 之前先刪掉舊的，如圖。

```
void CGameMap::RandomBouncingBall()
{
    const int MAX_RAND_NUM = 10;           //隨機的最大值
    random_num = (rand() % MAX_RAND_NUM)+1; //隨機1~MAX_RAND_NUM

    delete [] bballs;                      //先刪掉之前所配置的空間
    bballs = new CBouncingBall[random_num]; //根據隨機值動態配置CBounci
    int ini_index = 0;                     //初始化陣列索引
    for(int row=0; row<4; row++)
        for(int col=0; col<5; col++)
        {
            if(map[row][col] != 0 && ini_index < random_num) //只放球在有色的地圖且初始
            {
                InitializeBouncingBall(ini_index, row, col);
                ini_index++; //初始化的陣列索引加一
            }
        }
}
```

(c)但是，上述修改又會導致執行時出錯，這是因為當第一次按下時，**空間尚未被配置就執行刪除的動作(尚未new，就做delete)**。由於delete 指令只會對非NULL 的記憶體位址釋放記憶體空間。所以我們將bball 指標初始化為 NULL 即可解決問題，如下圖。

```
CGameMap::CGameMap()
:X(20), Y(40), MW(120), MH(100)
{
    int map_init[4][5] = {{0,0,1,0,0},
                          {0,1,2,1,0},
                          {1,2,1,2,1},
                          {2,1,2,1,2}};

    for(int i=0;i<4;i++)
    {
        for(int j=0;j<5;j++)
        {
            map[i][j] = map_init[i][j];
        }
    }

    random_num = 0; //初始化隨機個數
    bballs = NULL;
}
```