

國立臺北科技大學
2020 Spring 資工系物件導向程式實習
期末報告

Open Sonic



第 20 組

108590003 黃明萱

108590050 李浩銘

目錄

一、 簡介

1. 動機 2
2. 分工 2

二、 遊戲介紹

1. 遊戲說明 2
2. 遊戲圖形 3
3. 遊戲音效 4

三、 程式設計

1. 程式架構 5
2. 程式類別 7
3. 程式技術 8

四、 結語

1. 問題及解決方法 9
2. 時間表 11
3. 貢獻比例 11
4. 自我檢核表 12
5. 收獲 13
6. 心得、感想 14
7. 對於本課程的建議 14

一、 簡介

1.1 動機

我們一開始也在想要做什麼，就先回想小時候玩過了哪一些遊戲，原本有找了類似冰火兄弟的雙人遊戲，但好像說過不能選這類型的遊戲就沒有選擇了，最後想到音速小子這款遊戲，對我們來說都是兒時回憶，也有達到老師的遊戲標準，所以就想玩自己寫出來的音速小子，來重現兒時回憶。

1.2 分工

李浩銘：主要遊戲程式的撰寫，除錯，以及封裝遊戲。

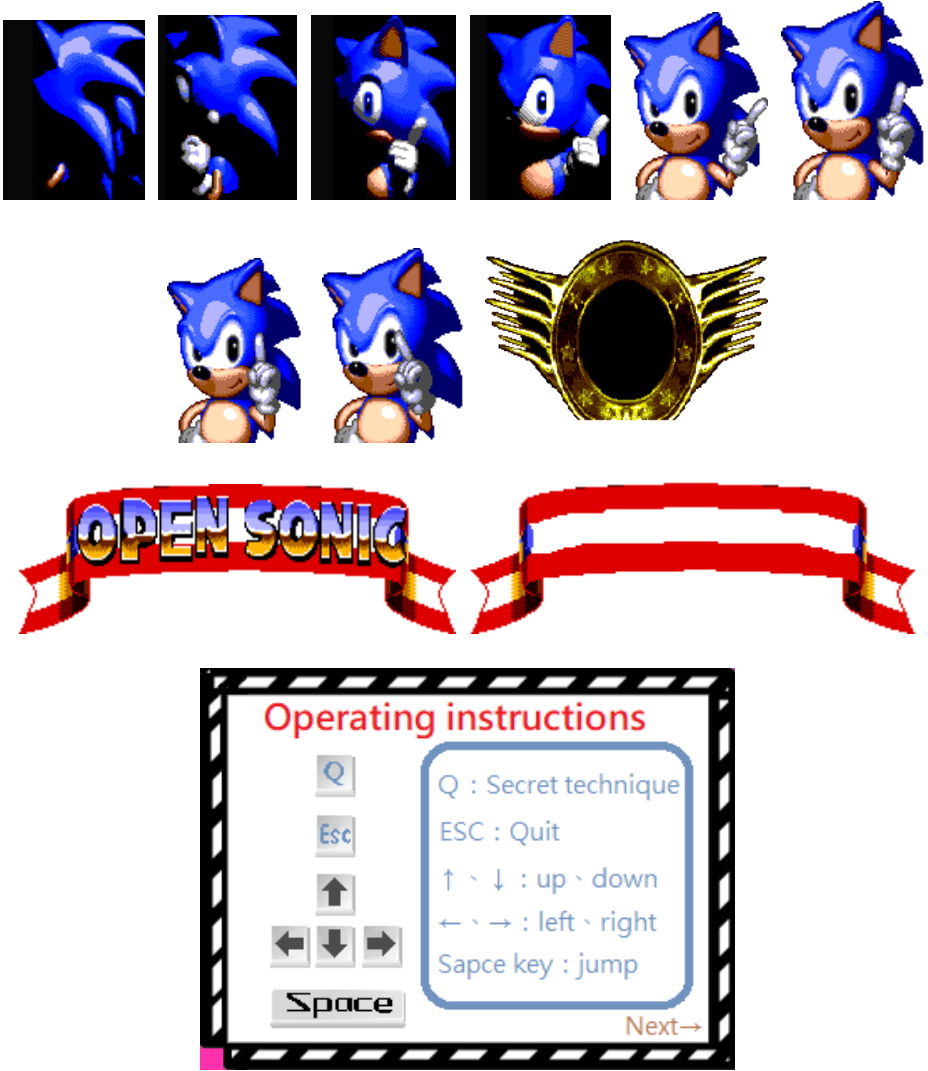
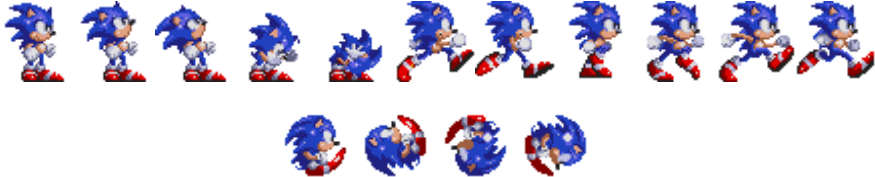

黃明萱：尋找、處理與裁減素材、部分遊戲程式的撰寫。

二、 遊戲介紹

2.1 遊戲說明

- (1) 遊戲操作：鍵盤上、下、左、右鍵控制角色方向的移動、空白鍵是跳躍。
- (2) 闖關模式：按照地圖跑到終點，會計時你花多久的時間，有金幣數量、分數。
- (3) 關卡陷阱：碰到怪物或踩到鋸齒都會game over，鋸齒會以三秒為一個循環出現。
- (4) 密技：鍵盤Q則可以隨意跑地圖。

2.2 遊戲圖形

進入遊戲動畫
 <p>The animation sequence for the game's intro. It begins with a series of six frames showing Sonic the Hedgehog in various poses, from a side profile to a full frontal view with his hands on his hips. This is followed by two more frames of Sonic in a similar pose. Then, a large, ornate golden ring with a black center and a star pattern appears. Below the ring, a red ribbon banner with the text "OPEN SONIC" in a stylized, colorful font is displayed. At the bottom, a black-bordered box with a white background contains the text "Operating instructions" in red. Below this text are several icons representing keyboard controls: a 'Q' key, an 'Esc' key, an up arrow, left, down, and right arrows, and a 'Space' key. To the right of these icons, a list of instructions is provided: "Q : Secret technique", "ESC : Quit", "↑、↓ : up、down", "←、→ : left、right", and "Space key : jump". A "Next→" button is located at the bottom right of the instructions box.</p>
遊戲角色：Sonic
 <p>A collection of Sonic the Hedgehog character sprites. The top row features ten different poses of Sonic, including walking, running, and jumping. The bottom row shows four smaller sprites of Sonic in various poses, including a crouching position.</p>
遊戲金幣、分數
 <p>Two items related to the game's scoring system. On the left is a small, yellow, circular gold coin. On the right is a larger, golden ring with a black center and a star pattern, which is used to represent the score.</p>
遊戲過關


怪物圖形

每一關物件

遊戲的英文

遊戲的Icon


2.3 遊戲音效

音樂檔案名稱	說明
title.mp3	遊戲進入選單畫面的音效
options.mp3	進入選單option有不同的音效
choose.wav	選單移動時的音效
select.wav	選單確認時的音效
return.wav	Esc退出的音效
crusader.mp3	每一關遊戲背景音效
ring.wav	遊戲裡吃金幣的音效

3.1 程式架構

我們把程式分成幾個主要的 class，分別是 Actor：角色、Bricks：方塊、Enemies：怪物、Items：物品、Level：關卡。

Actor：

這個類別是角色的 base-class，俱有 *angle* 角度、*pos* 位置、*velocity* 加速度、*delta* 位移等主要變數，以及一個指向類別 Bricks 的指標 *refBrick*。

在 *OnInit* 階段，載入各種動作的圖檔，並且設定座標。

在 *OnMove* 階段，把 *delta* 位移計算出來，在 3.3 會詳述。

在 *OnShow* 階段，根據玩家的操縱顯示各種動作。

Brick：

這個類別俱有 *id*、*angle* 角度、*property* 屬性、*behavior* 行為、*texture* 材質、*pos* 位置、*delta* 位移等主要變數。

在 *OnInit* 階段，會根據當前的關卡，以及 object 的 *id*，自動取得 *texture* 材質、*angle* 角度、*property* 屬性以及 *behavior* 行為。

在 *OnMove* 階段，會根據 *delta* 位移來進行移動。

在 *OnShow* 階段，會把 object 的 *texture* 材質顯示出來。

Enemy：

這個類別是怪物的 base-class，俱有 *speed* 速度、*direction* 方向、*texture* 材質、*spawn* 出生點、*pos* 位置、*delta* 位移等主要變數，以及一個指向類別 Bricks 的指標 *refBrick*。共有 17 個 derived-class。

在 *OnInit* 階段，載入相應的圖檔到 *texture* 材質，並且設定 *spawn* 出生點，根據 *spawn* 出生點找出 *refBrick*，再根據 *refBrick* 微調 *spawn* 出生點。

在 *OnMove* 階段，會根據 *delta* 位移來進行移動，然後再根據各個怪物的移動行為進行移動。

在 *OnShow* 階段，會根據各個怪物的顯示行為把 *texture* 材質顯示出來。

Item：

這個類別是物品的 base-class，俱有 *id*、*texture* 材質、*pos* 位置、*delta* 位移等主要變數，以及一個指向類別 Actor 的指標 *currentActor*。共有 8 個 derived-class。

在 *OnInit* 階段，載入相應的圖檔到 *texture* 材質，並且設定座標。

在 *OnMove* 階段，會根據 *delta* 位移來進行移動。

在 *OnShow* 階段，會根據各個物品的顯示行為把 *texture* 材質顯示出來。

Level :

這個類別是玩家與遊戲中各個 object 互動的橋樑，俱有 *rings* 戒指、*score* 分數、*delta* 位移等主要變數，並且有該關卡的 Actor 角色、所有 Brick 方塊、所有 Item 物品以及所有 Enemy 怪物的指標。

在 *OnInit* 階段，根據所選擇的關卡，載入該關卡的 Actor 角色、Brick 方塊、Item 物品以及 Enemy 怪物。

在 *OnMove* 階段，根據 Actor 的計算出來 *delta* 位移，控制 Brick 方塊、Item 物品以及 Enemy 怪物進行移動。

在 *OnShow* 階段，控制 Brick 方塊、Item 物品以及 Enemy 怪物把 *texture* 材質顯示出來。



3.2 程式類別

類別名稱	.h 檔行數	.cpp 檔行數	說明
Actor	176	657	角色的 Base Class
Sonic	/	/	角色 1
Miles	/	/	角色 2 (Deprecated 棄用)
Knuckles	/	/	角色 3 (Deprecated 棄用)
Brick	73	1288	方塊
Enemy	246	761	敵人的 Base Class
Bamboo	/	/	敵人 1
Chef	/	/	敵人 2
Earthworm	/	/	敵人 3
Fly	/	/	敵人 4
Groundhog	/	/	敵人 5
Insect	/	/	敵人 6
Ladybug	/	/	敵人 7
Mosquito	/	/	敵人 8
Penguin	/	/	敵人 9
PinkMoster	/	/	敵人 10
Rocket	/	/	敵人 11
Shark	/	/	敵人 12
Snails	/	/	敵人 13
Spider	/	/	敵人 14
Blue	/	/	敵人 15
FlyShield	/	/	敵人 16
Red	/	/	敵人 17
Item	729	1554	物品的 Base Class
Ring	/	/	物品 1
BIG_Ring	/	/	物品 2
GATE_YELLOW	/	/	物品 3
SEND	/	/	物品 4
SPIKES_UP	/	/	物品 5
SPIKES_DOWN	/	/	物品 6
SPIKES_LEFT	/	/	物品 7
SPIKES_RIGHT	/	/	物品 8
Level	71	697	關卡
總行數	1298	4950	6248

3.3 程式技術

我們參考了原 framework 中提供的 CInteger 類別，加入了一個名為 CString 的類別(.h 檔共 20 行，.cpp 檔共 120 行)，用以顯示字串圖形，原因在 4.1 問題三中詳述。算法如下：

```
int nx = x;
int len = s.length();

for (int i = 0; i < len; i++) {
    if (s.at(i) == ' ') // Handling space
        nx += (int)(factor * alphabet[0].Width());
    else {
        int alphabet_num = int(s.at(i)) - int('A'); // Encode input alphabet ASCII to number 0-26
        if (isFocus()) // Handling focus, yellow alphabet
            alphabet_num += 26;

        alphabet[alphabet_num].SetTopLeft(nx, y);
        alphabet[alphabet_num].ShowBitmap();
        nx += (int)(factor * alphabet[alphabet_num].Width());
    }
}
```

角色的移動應該可以說是整個遊戲中最複雜的部分了，其分成四個階段：一檢查、二 X 軸移動，三 Y 軸移動，四計算 *delta* 位移。在一檢查階段，會檢查經過上一次的 *delta* 位移後，角色有沒有觸碰到怪物或陷阱。

在二 X 軸移動階段，會計算 X 軸移動的 *acceleration* 加速度及 *velocity* 速度，這裡我們使用比較簡單的 constant acceleration，設定 acceleration 為 4，公式為 $v = u + at$ ，轉換成程式碼即為 $velocity.x += acceleration$ ，之後在加上 friction 摩擦力以及 drag 風阻作為減速，因為這裡涉及到比較複雜物理計算，因此我們把減速簡化成一個少於 1 的係數。

在三 Y 軸移動階段，會計算 Y 軸移動的 *acceleration* 加速度及 *velocity* 速度，這裡我們使用自由落體的公式 $v = gt$ ，轉換成程式碼即為 $velocity.y += gravity$ ，因為我們簡化了 friction 摩擦力以及 drag 風阻，因此 terminal velocity 也簡化成一個常數。對於跳躍的部分，角色有一個 jumpStrength，用於處理跳躍時 $velocity.y -= jumpStrength$ 。

在四計算 *delta* 位移階段，會根據階段二和階段三計算出來的 velocity，以以下的公式的計算出 *delta* 位移，level 會取得這個 *delta* 位移，然後傳入各個 Brick，Item，和 Enemy，把角色 *delta* 位移的轉換成鏡頭的 *delta* 位移。

$$\begin{aligned}\Delta x &= velocity_x \cdot \cos(angle) + velocity_y \cdot \sin(angle) \\ \Delta y &= velocity_y \cdot \cos(angle) - velocity_x \cdot \sin(angle)\end{aligned}$$

4.1 問題及解決方法

問題一：

原本遊戲內的背景是從右往左無限跑的動畫，並且分成不同的 Layer，速度會有不一樣。

解決方法：

我們嘗試設定圖片跑到某個位置會重置，無果。最後放棄使用動畫，改成一張背景圖片。

問題二：

開源文件中只提供了各個物品、方塊、角色及怪物連續圖，但是 framework 卻要一張一張地載入。

解決方法：

也沒辦法只好一張一張把圖片剪出來，再放進遊戲中。

問題三：

開源文件中只提供了 A~Z 的字母圖片，我們須要想辦法把字母圖片組合成字串顯示出來。

解決方法：

我們想到 2 種方法，一是把字母圖片合併成我們想要顯示的字串。二是利用程式解決。我們選擇了後者，因為這樣的自由度更高，實作起來也比較快速。

問題四：

關於斜坡，基於這個 Framework 中的 Bounding Box 跟圖片的大小相同，我們沒辦法做出三角形的 Bounding Box，讓人物跟著 Bounding Box 去移動。

解決方法：

在類別 *Bricks* 中加入變數 *angle*，通過公式 $\Delta y = velocity_x \cdot \tan(angle)$ 去計算出 Y 軸的相對移動距離 Δy 。

問題五：

承接上一個問題，關於 180 度迴轉，根據上述的計算方法，在變數 *angle* 等於 90 度時，Y 軸的相對位移 Δy 理論上會等於無限。我們也有嘗試修改成原作者的計算方法，如下：

$$\Delta x = velocity_x \cdot \cos(angle) + velocity_y \cdot \sin(angle)$$

$$\Delta y = velocity_y \cdot \cos(angle) - velocity_x \cdot \sin(angle)$$

修改計算方法後發現 *angle* 等於 90 度時，Y 軸的相對位移 Δy 會等於 0。最後發現是原作者的判斷方法跟我們的不一樣所導致的。

解決方法：

我們參考了原作者的判斷方法，嘗試進行修改。加角色身上加入一個判斷點，當 0 度時，判斷點在下方；當大於 0 度及小於 90 度，判斷點在右下方；當 90 度時，判斷點在右方；當大於 90 度及小於 180 度，判斷點在右上方；當 180 度時，判斷點在上方；當大於 180 度及小於 270 度，判斷點在左上方；當 270 度時，判斷點在左方；當大於 270 度及小於 360 度，判斷點在左下方。

問題六：

承接上一個問題，關於 180 度迴轉，這個判斷方法的確能夠通過部分 180 度迴轉。但是我們很快就發現了這個判斷方法並不完整，導致無法通過所有 180 度迴轉。經過很長時間的思考，計算以及修改後，我們還是無法解決判斷方法不完整的問題。

解決方法：

最後我們自己寫了一個比較不好判斷方法，就是在 180 度迴轉的不同位置加入不同的透明點，角色在通過這幾個透明點的時候，會改變移動的角度。雖然這並非一個很好的解決方法，但是至少可以成功實作出來。

問題七：

關於 360 度迴轉，我們發現在 360 度迴轉的時候，角色無法離開 360 度迴轉，原因是角色在碰到 360 度迴轉的最後一個透明點的下一個 frame，又會再碰到 360 度迴轉的第一個透明點而再次進入 360 度迴轉，造成角色無法離開 360 度迴轉。

解決方法：

我們在第一個透明點上增加一點判斷，令角色再最後一個透明點回來時不會再次進入 360 度迴轉。

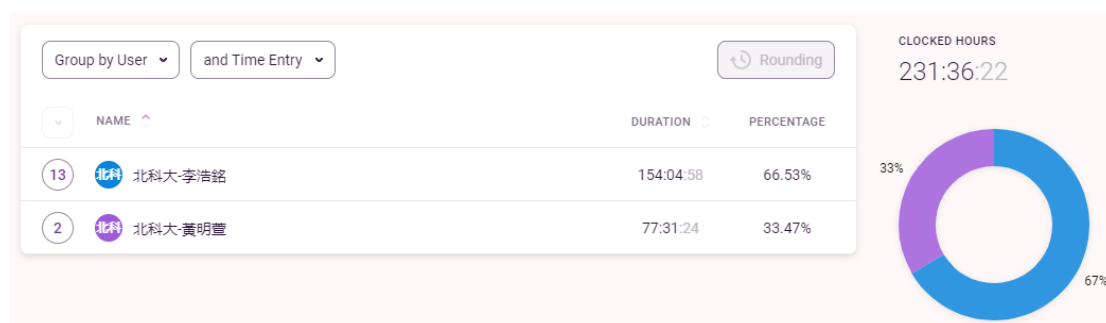
4.2 時間表 W

週次	108590003	108590050	說明
1 02/26-03/04	0	0	選定遊戲
2 03/05-03/11	0	0	完成 tutorial
3 03/12-03/18	7	13	理解 framework
4 03/19-03/25	2	15	完成主畫面
5 03/26-04/01	1	8	完成各個選單的內容
6 04/02-04/08	4	21	完成各個類別的 base-class
7 04/09-04/15	1	8	完成 Enemy 的各個 derived-class
8 04/16-04/22	5	4	完成類別 Level
9 04/23-04/29	2	2	完成類別 Item 的各個 derived-class
10 04/30-05/06	0	6	完成 Actor 的各個 derived-class
11 05/07-05/13	16	16	完善 Level 對各個 object 的控制
12 05/14-05/20	0	5	研究角色移動物理
13 05/21-05/27	2	8	研究角色移動物理
14 05/28-06/03	0	10	處理角色在迴轉時的問題
15 06/04-06/10	8	15	處理角色在迴轉時的問題
16 06/11-06/17	11	11	完成計分，計時，音效
17 06/18-06/25	14	14	完成通關以及死亡畫面
總時數	78	154	232

4.3 貢獻比例

黃明萱：78 小時，33.47%

李浩銘：154 小時，66.53%



4.4 自我檢核表

週次	項目	完成否	無法完成原因
1	解決 Memory leak	<input type="checkbox"/> 已完成 <input checked="" type="checkbox"/> 未完成	已嘗試解決 Memory leak，但是程式出現錯誤，無法解決
2	自定遊戲 Icon	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
3	全螢幕啟動	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
4	有 About 畫面	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
5	初始畫面說明按鍵及滑鼠之用法與密技	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
7	上傳 setup/apk/source 檔	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
8	setup 檔可正確執行	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
9	報告字型、點數、對齊、行距、頁碼等格式正確	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	

4.5 收獲

黃明萱：

上學期在熟悉語法，這學期直接製作，這次程式使用了不少 virtual function、Polymorphism、物理計算，讓我知道物理很重要，但好險可以 google 找到相關的物理公式，把所學到的知識全部結合在實作上面，能更加的了解程式的運作，也讓我除了程式碼以外，遊戲畫面裡的各種角色、地圖的構成、機關都是用圖片呈現出來的，所以處理圖片也是相當花費時間，雖然我們有找到相關資源，但是裡面的圖片一張內有好幾個物件圖，必須把圖片一個一個進行裁剪，有些動畫分解圖，運用到 CAnimation 來呈現出更好的效果。除此之外也讓我更了解有些想要呈現的東西應該要寫在哪個區塊，怎麼寫判斷才會觸發到，不斷思考與隊員討論怎樣才會更好，讓遊戲呈現出我要的結果。

李浩銘：

經過一個學期的實習後，我學到了不少東西，當中最主要的當然是了解了遊戲 framework 的運作，從整個遊戲的生命週期，以至於每一個 stage 的生命週期，老師提供的這個遊戲 framework 老師提供的這個 framework 都跟外面其他的 framework 大同小異。

在程式的文法上，我學到了一個新的東西叫 Enum Class，因為我在撰寫程式碼的時候有用到幾個 enum，但是有部分 enum 內的成員有重複，查了一下之後發現可以用 Enum Class 來解決。除此之外，這個學期的實習還讓我更深入地了解了物品導向，以及繼承等的觀念。

在除錯方式上，除了傳統的除錯方法，把變數打印出來，我還學到了如何使用除錯器。通過在程式碼中設定 break point，可以暫停程式的運行，並查看記憶體空間等。

4.6 心得與感想

黃明萱：

剛開始真的是很怕寫不出來，雖然上學期的課程有學到了 C++ 程式語言的架構，但是有些地方真的還是會不清楚，也知道自己的程式基礎沒有很好，而且更害怕拖累到隊友。剛開始的時候，比較簡單的東西隊友都先交給我像是程式裡的 Item、Brick 的 virtual function、Polymorphism，我覺得還行運用之前學得，但隨著東西越來越多，有些東西就只能與隊友討論或看有什麼解決方法，我們遇到最大的困難真的是那些角度，用了超多方法，都會穿牆，最後終於搞懂了。這次修物件導向程式設計實習課，讓我知道遊戲裡中的角色、物件、機關都很重要，定義方式也要很明確，當然最重要的是團隊合作的力量，我真的很感謝隊友因為常常我會自己卡住，所以我也從他身上學到很多。

李浩銘：

我以前有寫過小遊戲的經驗，想說 OOPL 要寫遊戲，應該還好啦。結果選了一個讓自己很後悔的遊戲。一來是遊戲的東西很多，林林總總的方塊，物品，怪物。二來是 framework 的功能不太完整，比如圖片的旋轉跟反轉都沒有，讓我花了很多時間去研究到底要怎樣實作出來。寫到後面的時候，看到其他組的作品，發現我們的進度都比其他組落後，覺得自己有可能完成不了遊戲，很害怕被當掉。這個時候我還卡在角色迴轉，寫了兩個星期還是寫不出，真的感覺很迷茫，心很累，一度想放棄，但是我的組員把我從放棄的邊緣拉了回來，真的很感謝她。最後我們砍掉了很多功能，把迴轉硬幹出來，勉強地把遊戲完成了。

4.7 對於本課程的建議

Framework 增加旋轉圖片及反轉圖片功能。

5 附錄

gamelib.h

```

14 #define DEFAULT_BG_ALPHA RGB(255,0,255) // Set the default Background Alpha
15 to (255,0,255)
16 #define DEFAULT_SCALE SIZE_X/320
17 #define M_PI 3.14159265358979323846
18 ///////////////////////////////////////////////////
19 // 這個class提供顯示字串圖形的能力
20 // 每個Public Interface的用法都要懂，Implementation可以不懂
21 ///////////////////////////////////////////////////
22 class CString {
23 public:
24     CString();
25     CMovingBitmap *GetAlphabet(); // 回傳字母
26     void LoadBitmap(); // 載入A..Z之圖形
27     void SetTopLeft(int, int); // 將動畫的左上角座標移至 (x,y)
28     void ShowBitmap(string, double factor = DEFAULT_SCALE); // 將動畫貼到螢幕
29     bool isFocus();
30     void SetFocus(bool);
31 private:
32     static CMovingBitmap alphabet[52]; // 儲存A..Z之圖形(bitmap)
33     int x, y; // 顯示的座標
34     bool focus;
35     bool isBmpLoaded; // 是否已經載入圖形
36 };

```

gamelib.cpp

```

344 // CString: 這個class提供顯示字串圖形的能力
345 // 1. 要懂得怎麼呼叫(運用)其各種能力，但是可以不懂下列的程式是什麼意思
346 // 2. 自己寫到運用CMovingBitmap的程式時，可以參考下列程式的寫法
347 ///////////////////////////////////////////////////
348 CMovingBitmap CString::alphabet[52];
349 CString::CString()
350 {
351     isBmpLoaded = false;
352     focus = false;
353 }
354 CMovingBitmap* CString::GetAlphabet() {
355     return alphabet;
356 }
357 void CString::LoadBitmap()
358 {
359     //
360     // digit[i]為class varibale，所以必須避免重複LoadBitmap
361     //
362     if (!isBmpLoaded) {
363         int d[52] = {
364             FONT_WHITE_SHADOW_UPPER_A,
365             FONT_WHITE_SHADOW_UPPER_B,
366             FONT_WHITE_SHADOW_UPPER_C,
367             FONT_WHITE_SHADOW_UPPER_D,
368             FONT_WHITE_SHADOW_UPPER_E,
369             FONT_WHITE_SHADOW_UPPER_F,
370             FONT_WHITE_SHADOW_UPPER_G,
371             FONT_WHITE_SHADOW_UPPER_H,
372             FONT_WHITE_SHADOW_UPPER_I,
373             FONT_WHITE_SHADOW_UPPER_J,
374             FONT_WHITE_SHADOW_UPPER_K,
375             FONT_WHITE_SHADOW_UPPER_L,
376             FONT_WHITE_SHADOW_UPPER_M,
377             FONT_WHITE_SHADOW_UPPER_N,
378             FONT_WHITE_SHADOW_UPPER_O,
379             FONT_WHITE_SHADOW_UPPER_P,
380             FONT_WHITE_SHADOW_UPPER_Q,
381             FONT_WHITE_SHADOW_UPPER_R,
382             FONT_WHITE_SHADOW_UPPER_S,
383             FONT_WHITE_SHADOW_UPPER_T,
384             FONT_WHITE_SHADOW_UPPER_U,
385             FONT_WHITE_SHADOW_UPPER_V,
386             FONT_WHITE_SHADOW_UPPER_W,
387             FONT_WHITE_SHADOW_UPPER_X,
388             FONT_WHITE_SHADOW_UPPER_Y,
389             FONT_WHITE_SHADOW_UPPER_Z,
390             FONT_WHITE_SHADOW_LOWER_A,
391             FONT_WHITE_SHADOW_LOWER_B,
392             FONT_WHITE_SHADOW_LOWER_C,
393             FONT_WHITE_SHADOW_LOWER_D,
394             FONT_WHITE_SHADOW_LOWER_E,
395             FONT_WHITE_SHADOW_LOWER_F,
396             FONT_WHITE_SHADOW_LOWER_G,
397             FONT_WHITE_SHADOW_LOWER_H,
398             FONT_WHITE_SHADOW_LOWER_I,
399             FONT_WHITE_SHADOW_LOWER_J,
400             FONT_WHITE_SHADOW_LOWER_K,
401             FONT_WHITE_SHADOW_LOWER_L,
402             FONT_WHITE_SHADOW_LOWER_M,
403             FONT_WHITE_SHADOW_LOWER_N,
404             FONT_WHITE_SHADOW_LOWER_O,
405             FONT_WHITE_SHADOW_LOWER_P,
406             FONT_WHITE_SHADOW_LOWER_Q,
407             FONT_WHITE_SHADOW_LOWER_R,
408             FONT_WHITE_SHADOW_LOWER_S,
409             FONT_WHITE_SHADOW_LOWER_T,
410             FONT_WHITE_SHADOW_LOWER_U,
411             FONT_WHITE_SHADOW_LOWER_V,
412             FONT_WHITE_SHADOW_LOWER_W,
413             FONT_WHITE_SHADOW_LOWER_X,
414             FONT_WHITE_SHADOW_LOWER_Y,
415             FONT_WHITE_SHADOW_LOWER_Z,
416             FONT_WHITE_SHADOW_UPPER_0,
417             FONT_WHITE_SHADOW_UPPER_1,
418             FONT_WHITE_SHADOW_UPPER_2,
419             FONT_WHITE_SHADOW_UPPER_3,
420             FONT_WHITE_SHADOW_UPPER_4,
421             FONT_WHITE_SHADOW_UPPER_5,
422             FONT_WHITE_SHADOW_UPPER_6,
423             FONT_WHITE_SHADOW_UPPER_7,
424             FONT_WHITE_SHADOW_UPPER_8,
425             FONT_WHITE_SHADOW_UPPER_9,
426             FONT_WHITE_SHADOW_LOWER_0,
427             FONT_WHITE_SHADOW_LOWER_1,
428             FONT_WHITE_SHADOW_LOWER_2,
429             FONT_WHITE_SHADOW_LOWER_3,
430             FONT_WHITE_SHADOW_LOWER_4,
431             FONT_WHITE_SHADOW_LOWER_5,
432             FONT_WHITE_SHADOW_LOWER_6,
433             FONT_WHITE_SHADOW_LOWER_7,
434             FONT_WHITE_SHADOW_LOWER_8,
435             FONT_WHITE_SHADOW_LOWER_9,
436             FONT_WHITE_SHADOW_UPPER_SPACE,
437             FONT_WHITE_SHADOW_LOWER_SPACE,
438             FONT_WHITE_SHADOW_UPPER_HASH,
439             FONT_WHITE_SHADOW_LOWER_HASH,
440             FONT_WHITE_SHADOW_UPPER_AMP,
441             FONT_WHITE_SHADOW_LOWER_AMP,
442             FONT_WHITE_SHADOW_UPPER_AT,
443             FONT_WHITE_SHADOW_LOWER_AT,
444             FONT_WHITE_SHADOW_UPPER_ASTERISK,
445             FONT_WHITE_SHADOW_LOWER_ASTERISK,
446             FONT_WHITE_SHADOW_UPPER_CARET,
447             FONT_WHITE_SHADOW_LOWER_CARET,
448             FONT_WHITE_SHADOW_UPPER_DOLLAR,
449             FONT_WHITE_SHADOW_LOWER_DOLLAR,
450             FONT_WHITE_SHADOW_UPPER_EQUAL,
451             FONT_WHITE_SHADOW_LOWER_EQUAL,
452             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
453             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
454             FONT_WHITE_SHADOW_UPPER_HASH,
455             FONT_WHITE_SHADOW_LOWER_HASH,
456             FONT_WHITE_SHADOW_UPPER_LEFT_BRACKET,
457             FONT_WHITE_SHADOW_LOWER_LEFT_BRACKET,
458             FONT_WHITE_SHADOW_UPPER_RIGHT_BRACKET,
459             FONT_WHITE_SHADOW_LOWER_RIGHT_BRACKET,
460             FONT_WHITE_SHADOW_UPPER_LEFT_BRACE,
461             FONT_WHITE_SHADOW_LOWER_LEFT_BRACE,
462             FONT_WHITE_SHADOW_UPPER_RIGHT_BRACE,
463             FONT_WHITE_SHADOW_LOWER_RIGHT_BRACE,
464             FONT_WHITE_SHADOW_UPPER_COMMA,
465             FONT_WHITE_SHADOW_LOWER_COMMA,
466             FONT_WHITE_SHADOW_UPPER_PERIOD,
467             FONT_WHITE_SHADOW_LOWER_PERIOD,
468             FONT_WHITE_SHADOW_UPPER_SLASH,
469             FONT_WHITE_SHADOW_LOWER_SLASH,
470             FONT_WHITE_SHADOW_UPPER_SEMICOLON,
471             FONT_WHITE_SHADOW_LOWER_SEMICOLON,
472             FONT_WHITE_SHADOW_UPPER_SINGLE_QUOTE,
473             FONT_WHITE_SHADOW_LOWER_SINGLE_QUOTE,
474             FONT_WHITE_SHADOW_UPPER_DOUBLE_QUOTE,
475             FONT_WHITE_SHADOW_LOWER_DOUBLE_QUOTE,
476             FONT_WHITE_SHADOW_UPPER_TILDE,
477             FONT_WHITE_SHADOW_LOWER_TILDE,
478             FONT_WHITE_SHADOW_UPPER_UNDERSCORE,
479             FONT_WHITE_SHADOW_LOWER_UNDERSCORE,
480             FONT_WHITE_SHADOW_UPPER_CIRCUMFLEX,
481             FONT_WHITE_SHADOW_LOWER_CIRCUMFLEX,
482             FONT_WHITE_SHADOW_UPPER_GRAVE_ACCENT,
483             FONT_WHITE_SHADOW_LOWER_GRAVE_ACCENT,
484             FONT_WHITE_SHADOW_UPPER_CARET,
485             FONT_WHITE_SHADOW_LOWER_CARET,
486             FONT_WHITE_SHADOW_UPPER_DOLLAR,
487             FONT_WHITE_SHADOW_LOWER_DOLLAR,
488             FONT_WHITE_SHADOW_UPPER_EQUAL,
489             FONT_WHITE_SHADOW_LOWER_EQUAL,
490             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
491             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
492             FONT_WHITE_SHADOW_UPPER_HASH,
493             FONT_WHITE_SHADOW_LOWER_HASH,
494             FONT_WHITE_SHADOW_UPPER_AMP,
495             FONT_WHITE_SHADOW_LOWER_AMP,
496             FONT_WHITE_SHADOW_UPPER_AT,
497             FONT_WHITE_SHADOW_LOWER_AT,
498             FONT_WHITE_SHADOW_UPPER_ASTERISK,
499             FONT_WHITE_SHADOW_LOWER_ASTERISK,
500             FONT_WHITE_SHADOW_UPPER_CARET,
501             FONT_WHITE_SHADOW_LOWER_CARET,
502             FONT_WHITE_SHADOW_UPPER_DOLLAR,
503             FONT_WHITE_SHADOW_LOWER_DOLLAR,
504             FONT_WHITE_SHADOW_UPPER_EQUAL,
505             FONT_WHITE_SHADOW_LOWER_EQUAL,
506             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
507             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
508             FONT_WHITE_SHADOW_UPPER_HASH,
509             FONT_WHITE_SHADOW_LOWER_HASH,
510             FONT_WHITE_SHADOW_UPPER_AMP,
511             FONT_WHITE_SHADOW_LOWER_AMP,
512             FONT_WHITE_SHADOW_UPPER_AT,
513             FONT_WHITE_SHADOW_LOWER_AT,
514             FONT_WHITE_SHADOW_UPPER_ASTERISK,
515             FONT_WHITE_SHADOW_LOWER_ASTERISK,
516             FONT_WHITE_SHADOW_UPPER_CARET,
517             FONT_WHITE_SHADOW_LOWER_CARET,
518             FONT_WHITE_SHADOW_UPPER_DOLLAR,
519             FONT_WHITE_SHADOW_LOWER_DOLLAR,
520             FONT_WHITE_SHADOW_UPPER_EQUAL,
521             FONT_WHITE_SHADOW_LOWER_EQUAL,
522             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
523             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
524             FONT_WHITE_SHADOW_UPPER_HASH,
525             FONT_WHITE_SHADOW_LOWER_HASH,
526             FONT_WHITE_SHADOW_UPPER_AMP,
527             FONT_WHITE_SHADOW_LOWER_AMP,
528             FONT_WHITE_SHADOW_UPPER_AT,
529             FONT_WHITE_SHADOW_LOWER_AT,
530             FONT_WHITE_SHADOW_UPPER_ASTERISK,
531             FONT_WHITE_SHADOW_LOWER_ASTERISK,
532             FONT_WHITE_SHADOW_UPPER_CARET,
533             FONT_WHITE_SHADOW_LOWER_CARET,
534             FONT_WHITE_SHADOW_UPPER_DOLLAR,
535             FONT_WHITE_SHADOW_LOWER_DOLLAR,
536             FONT_WHITE_SHADOW_UPPER_EQUAL,
537             FONT_WHITE_SHADOW_LOWER_EQUAL,
538             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
539             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
540             FONT_WHITE_SHADOW_UPPER_HASH,
541             FONT_WHITE_SHADOW_LOWER_HASH,
542             FONT_WHITE_SHADOW_UPPER_AMP,
543             FONT_WHITE_SHADOW_LOWER_AMP,
544             FONT_WHITE_SHADOW_UPPER_AT,
545             FONT_WHITE_SHADOW_LOWER_AT,
546             FONT_WHITE_SHADOW_UPPER_ASTERISK,
547             FONT_WHITE_SHADOW_LOWER_ASTERISK,
548             FONT_WHITE_SHADOW_UPPER_CARET,
549             FONT_WHITE_SHADOW_LOWER_CARET,
550             FONT_WHITE_SHADOW_UPPER_DOLLAR,
551             FONT_WHITE_SHADOW_LOWER_DOLLAR,
552             FONT_WHITE_SHADOW_UPPER_EQUAL,
553             FONT_WHITE_SHADOW_LOWER_EQUAL,
554             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
555             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
556             FONT_WHITE_SHADOW_UPPER_HASH,
557             FONT_WHITE_SHADOW_LOWER_HASH,
558             FONT_WHITE_SHADOW_UPPER_AMP,
559             FONT_WHITE_SHADOW_LOWER_AMP,
560             FONT_WHITE_SHADOW_UPPER_AT,
561             FONT_WHITE_SHADOW_LOWER_AT,
562             FONT_WHITE_SHADOW_UPPER_ASTERISK,
563             FONT_WHITE_SHADOW_LOWER_ASTERISK,
564             FONT_WHITE_SHADOW_UPPER_CARET,
565             FONT_WHITE_SHADOW_LOWER_CARET,
566             FONT_WHITE_SHADOW_UPPER_DOLLAR,
567             FONT_WHITE_SHADOW_LOWER_DOLLAR,
568             FONT_WHITE_SHADOW_UPPER_EQUAL,
569             FONT_WHITE_SHADOW_LOWER_EQUAL,
570             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
571             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
572             FONT_WHITE_SHADOW_UPPER_HASH,
573             FONT_WHITE_SHADOW_LOWER_HASH,
574             FONT_WHITE_SHADOW_UPPER_AMP,
575             FONT_WHITE_SHADOW_LOWER_AMP,
576             FONT_WHITE_SHADOW_UPPER_AT,
577             FONT_WHITE_SHADOW_LOWER_AT,
578             FONT_WHITE_SHADOW_UPPER_ASTERISK,
579             FONT_WHITE_SHADOW_LOWER_ASTERISK,
580             FONT_WHITE_SHADOW_UPPER_CARET,
581             FONT_WHITE_SHADOW_LOWER_CARET,
582             FONT_WHITE_SHADOW_UPPER_DOLLAR,
583             FONT_WHITE_SHADOW_LOWER_DOLLAR,
584             FONT_WHITE_SHADOW_UPPER_EQUAL,
585             FONT_WHITE_SHADOW_LOWER_EQUAL,
586             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
587             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
588             FONT_WHITE_SHADOW_UPPER_HASH,
589             FONT_WHITE_SHADOW_LOWER_HASH,
590             FONT_WHITE_SHADOW_UPPER_AMP,
591             FONT_WHITE_SHADOW_LOWER_AMP,
592             FONT_WHITE_SHADOW_UPPER_AT,
593             FONT_WHITE_SHADOW_LOWER_AT,
594             FONT_WHITE_SHADOW_UPPER_ASTERISK,
595             FONT_WHITE_SHADOW_LOWER_ASTERISK,
596             FONT_WHITE_SHADOW_UPPER_CARET,
597             FONT_WHITE_SHADOW_LOWER_CARET,
598             FONT_WHITE_SHADOW_UPPER_DOLLAR,
599             FONT_WHITE_SHADOW_LOWER_DOLLAR,
600             FONT_WHITE_SHADOW_UPPER_EQUAL,
601             FONT_WHITE_SHADOW_LOWER_EQUAL,
602             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
603             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
604             FONT_WHITE_SHADOW_UPPER_HASH,
605             FONT_WHITE_SHADOW_LOWER_HASH,
606             FONT_WHITE_SHADOW_UPPER_AMP,
607             FONT_WHITE_SHADOW_LOWER_AMP,
608             FONT_WHITE_SHADOW_UPPER_AT,
609             FONT_WHITE_SHADOW_LOWER_AT,
610             FONT_WHITE_SHADOW_UPPER_ASTERISK,
611             FONT_WHITE_SHADOW_LOWER_ASTERISK,
612             FONT_WHITE_SHADOW_UPPER_CARET,
613             FONT_WHITE_SHADOW_LOWER_CARET,
614             FONT_WHITE_SHADOW_UPPER_DOLLAR,
615             FONT_WHITE_SHADOW_LOWER_DOLLAR,
616             FONT_WHITE_SHADOW_UPPER_EQUAL,
617             FONT_WHITE_SHADOW_LOWER_EQUAL,
618             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
619             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
620             FONT_WHITE_SHADOW_UPPER_HASH,
621             FONT_WHITE_SHADOW_LOWER_HASH,
622             FONT_WHITE_SHADOW_UPPER_AMP,
623             FONT_WHITE_SHADOW_LOWER_AMP,
624             FONT_WHITE_SHADOW_UPPER_AT,
625             FONT_WHITE_SHADOW_LOWER_AT,
626             FONT_WHITE_SHADOW_UPPER_ASTERISK,
627             FONT_WHITE_SHADOW_LOWER_ASTERISK,
628             FONT_WHITE_SHADOW_UPPER_CARET,
629             FONT_WHITE_SHADOW_LOWER_CARET,
630             FONT_WHITE_SHADOW_UPPER_DOLLAR,
631             FONT_WHITE_SHADOW_LOWER_DOLLAR,
632             FONT_WHITE_SHADOW_UPPER_EQUAL,
633             FONT_WHITE_SHADOW_LOWER_EQUAL,
634             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
635             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
636             FONT_WHITE_SHADOW_UPPER_HASH,
637             FONT_WHITE_SHADOW_LOWER_HASH,
638             FONT_WHITE_SHADOW_UPPER_AMP,
639             FONT_WHITE_SHADOW_LOWER_AMP,
640             FONT_WHITE_SHADOW_UPPER_AT,
641             FONT_WHITE_SHADOW_LOWER_AT,
642             FONT_WHITE_SHADOW_UPPER_ASTERISK,
643             FONT_WHITE_SHADOW_LOWER_ASTERISK,
644             FONT_WHITE_SHADOW_UPPER_CARET,
645             FONT_WHITE_SHADOW_LOWER_CARET,
646             FONT_WHITE_SHADOW_UPPER_DOLLAR,
647             FONT_WHITE_SHADOW_LOWER_DOLLAR,
648             FONT_WHITE_SHADOW_UPPER_EQUAL,
649             FONT_WHITE_SHADOW_LOWER_EQUAL,
650             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
651             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
652             FONT_WHITE_SHADOW_UPPER_HASH,
653             FONT_WHITE_SHADOW_LOWER_HASH,
654             FONT_WHITE_SHADOW_UPPER_AMP,
655             FONT_WHITE_SHADOW_LOWER_AMP,
656             FONT_WHITE_SHADOW_UPPER_AT,
657             FONT_WHITE_SHADOW_LOWER_AT,
658             FONT_WHITE_SHADOW_UPPER_ASTERISK,
659             FONT_WHITE_SHADOW_LOWER_ASTERISK,
660             FONT_WHITE_SHADOW_UPPER_CARET,
661             FONT_WHITE_SHADOW_LOWER_CARET,
662             FONT_WHITE_SHADOW_UPPER_DOLLAR,
663             FONT_WHITE_SHADOW_LOWER_DOLLAR,
664             FONT_WHITE_SHADOW_UPPER_EQUAL,
665             FONT_WHITE_SHADOW_LOWER_EQUAL,
666             FONT_WHITE_SHADOW_UPPER_EXCLAMATION,
667             FONT_WHITE_SHADOW_LOWER_EXCLAMATION,
668             FONT_WHITE_SHADOW_UPPER_HASH,
669             FONT_WHITE_SHADOW_LOWER_HASH,
670             FONT_WHITE_SHADOW_UPPER_AMP,
671             FONT_WHITE_SHADOW_LOWER_AMP,
672             FONT_WHITE_SHADOW_UPPER_AT,
673             FONT_WHITE_SHADOW_LOWER_AT,
674             FONT_WHITE_SHADOW_UPPER_ASTERISK,
675             FONT_WHITE_SHADOW_LOWER_ASTERISK,
676             FONT_WHITE_SHADOW_UPPER_CARET,
677             FONT_WHITE_SHADOW_LOWER_CARET,
678             FONT_WHITE_SHADOW_UPPER_DOLLAR,
679             FONT_WHITE_SHADOW_LOWER_DOLLAR,
680             FONT_WHITE_SHADOW_UPPER_EQUAL,
6
```



```

386 FONT_WHITE_SHADOW_UPPER_T,
387 FONT_WHITE_SHADOW_UPPER_U,
388 FONT_WHITE_SHADOW_UPPER_V,
389 FONT_WHITE_SHADOW_UPPER_W,
390 FONT_WHITE_SHADOW_UPPER_X,
391 FONT_WHITE_SHADOW_UPPER_Y,
392 FONT_WHITE_SHADOW_UPPER_Z,
394 FONT_YELLOW_SHADOW_UPPER_A,
395 FONT_YELLOW_SHADOW_UPPER_B,
396 FONT_YELLOW_SHADOW_UPPER_C,
397 FONT_YELLOW_SHADOW_UPPER_D,
398 FONT_YELLOW_SHADOW_UPPER_E,
399 FONT_YELLOW_SHADOW_UPPER_F,
400 FONT_YELLOW_SHADOW_UPPER_G,
401 FONT_YELLOW_SHADOW_UPPER_H,
402 FONT_YELLOW_SHADOW_UPPER_I,
403 FONT_YELLOW_SHADOW_UPPER_J,
404 FONT_YELLOW_SHADOW_UPPER_K,
405 FONT_YELLOW_SHADOW_UPPER_L,
406 FONT_YELLOW_SHADOW_UPPER_M,
407 FONT_YELLOW_SHADOW_UPPER_N,
408 FONT_YELLOW_SHADOW_UPPER_O,
409 FONT_YELLOW_SHADOW_UPPER_P,
410 FONT_YELLOW_SHADOW_UPPER_Q,
411 FONT_YELLOW_SHADOW_UPPER_R,
412 FONT_YELLOW_SHADOW_UPPER_S,
413 FONT_YELLOW_SHADOW_UPPER_T,
414 FONT_YELLOW_SHADOW_UPPER_U,
415 FONT_YELLOW_SHADOW_UPPER_V,
416 FONT_YELLOW_SHADOW_UPPER_W,
417 FONT_YELLOW_SHADOW_UPPER_X,
418 FONT_YELLOW_SHADOW_UPPER_Y,
419 FONT_YELLOW_SHADOW_UPPER_Z,
421 };
422 for (int i = 0; i < 52; i++)
423     alphabet[i].LoadBitmap(d[i]);
424 isBmpLoaded = true;
425 }
426 }
427
428 void CString::SetTopLeft(int nx, int ny) // 將動畫的左上角座標移至 (x,y)
429 {
430     x = nx; y = ny;
431 }
432
433 bool CString::isFocus() {
434     return focus;
435 }
436
437 void CString::SetFocus(bool _focus) {
438     focus = _focus;
439 }
440
441 void CString::ShowBitmap(string s, double factor)
442 {
443     GAME_ASSERT(isBmpLoaded, "CString: 請先執行LoadBitmap，然後才能
444     ShowBitmap");
445     int nx = x;
446     int len = s.length();
447     for (int i = 0; i < len; i++) {
448         if (s.at(i) == ' ') // Handling space
449             nx += (int)(factor * alphabet[0].Width());
450         else {
451             int alphabet_num = int(s.at(i)) - int('A'); // Encode input alphabet ASCII
452             to number 0~26
453             if (isFocus()) // Handling focus, yellow alphabet
454                 alphabet_num += 26;
455             alphabet[alphabet_num].SetTopLeft(nx, y);
456             alphabet[alphabet_num].ShowBitmap();
457             nx += (int)(factor * alphabet[alphabet_num].Width());
458         }
459     }
460 }
461 }
462

```

```

mygame.h
#include "Level.h"
#include <cmath>
class Level;
namespace game_framework {
    ////////////////////////////////////////////////////
    // Constants
    ////////////////////////////////////////////////////
    enum AUDIO_ID { // 定義各種音效的編號
        AUDIO_1UP, // 0
        AUDIO_ACIDSHIELD, // 1
        AUDIO_BIGRING, // 2
        AUDIO_BIGSHOT, // 3
        AUDIO_BOSS, // 4
        AUDIO_BOSSHIT, // 5
        AUDIO_BRAKE, // 6
        AUDIO BUMPER, // 7
        AUDIO_CASH, // 8
        AUDIO_CHECKPOINT, // 9
        AUDIO_CHOOSE, // 10
        AUDIO_CRUSADER, // 11
        AUDIO_DEATH, // 12
        AUDIO_DENY, // 13
        AUDIO_DESTROYPOP, // 14
        AUDIO_DOOR1, // 15
        AUDIO_DOOR2, // 16
        AUDIO_ENDSIGN, // 17
        AUDIO_FIRE, // 18
        AUDIO_FIRE2, // 19
        AUDIO_FIRESHIELD, // 20
        AUDIO_FLOORFALL, // 21
        AUDIO_GLASSES, // 22
        AUDIO_GOAL, // 23
        AUDIO_INVINCIBLE, // 24
        AUDIO_JUMP, // 25
        AUDIO_LUMINOUS_VEIN, // 26
        AUDIO_OPTIONS, // 27
        AUDIO_PUFF, // 28
        AUDIO_RETURN, // 29
        AUDIO_RING, // 30
        AUDIO_RINGCOUNT, // 31
        AUDIO_RINGLESS, // 32
        AUDIO_SCRATCH, // 33
        AUDIO_SELECT, // 34
        AUDIO_SHIELD, // 35
        AUDIO_SHOT, // 36
        AUDIO_SPEED, // 37
        AUDIO_SPIKES, // 38
        AUDIO_SPIKES_APPEARING, // 39
        AUDIO_SPIKES_DISAPPEARING, // 40
        AUDIO_SPIN, // 41
        AUDIO_SPINDASH1, // 42
        AUDIO_SPINDASH2, // 43
        AUDIO_SPRING, // 44
        AUDIO_SWITCH, // 45
        AUDIO_TELEPORTER, // 46
        AUDIO_TITLE, // 47
    };
    enum class MENU {
        START_GAME,
        TUTORIAL,
        EXTRAS,
        OPTION,
        EXIT,
        COUNT
    };
    enum class OPTION {
        FULLSCREEN,
        STAGE_SELECT,
    };
}

```

```

        BACK,
        COUNT
};
enum class OPTION_SELECT {
    NO,
    YES,
    COUNT,
};
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// 這個class為遊戲的遊戲開頭畫面物件
// 每個Member function的Implementation都要弄懂
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
class CGameStateInit : public CGameState {
public:
    CGameStateInit(CGame *g);
    ~CGameStateInit();
    void OnInit(); // 遊戲的初值及圖形設定
    void OnBeginState(); // 設定每次重玩所需的變數
    void OnKeyDown(UINT, UINT, UINT); // 處理鍵盤Up的動作
protected:
    void OnMove();
    void OnShow(); // 顯示這個狀態的遊戲畫面
private:
    int page;
    int current_select;
    int option_select;
    vector<int> option_sel;
    // Intro animate
    CAnimation intro;
    bool intro_done;
    CMovingBitmap img_tutro;
    bool tutro;
    // Background
    CMovingBitmap background1;
    CMovingBitmap background2;
    CMovingBitmap background3;
    CMovingBitmap background4;
    CMovingBitmap background5;
    CMovingBitmap optionBG;
    // Logo
    CAnimation logo; // Logo animate
    CMovingBitmap logo1; //
    CMovingBitmap logo2; //
    // Menu
    CString stringHandler;
    CInteger numberHandler = CInteger(4);
    vector<string> menu;
    // Option
    vector<string> option;
    vector<string> bin_select;
    vector<string> tri_select;
    // stages
    vector<string> stages;
    int optionStage;
    int stageSelect;
};
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡
// 每個Member function的Implementation都要弄懂
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
class CGameStateRun : public CGameState {
public:
    CGameStateRun(CGame *g);
    ~CGameStateRun();
    void OnInit(); // 遊戲的初值及圖形設定
    void OnBeginState(); // 設定每次重玩所需的變數
    void OnKeyDown(UINT, UINT, UINT);
    void OnKeyUp(UINT, UINT, UINT);
protected:

```

```

        void OnMove();           // 移動遊戲元素
        void OnShow();           // 顯示這個狀態的遊戲畫面
private:
    CString stringHandler;
    CMovingBitmap Score;
    CMovingBitmap Time;
    CMovingBitmap Rings;
    CMovingBitmap colon;
    CInteger score = CInteger(4);
    CInteger rings = CInteger(3);
    CInteger t_min = CInteger(2);
    CInteger t_sec = CInteger(2);
    Level *level;
    int current_actor;
    int current_level;
    int game_time;
    bool debugMODE;
};
// 這個class為遊戲的結束狀態(Game Over)
// 每個Member function的Implementation都要弄懂
// 這個class為遊戲的結束狀態(Game Over)
class CGameStateOver : public CGameState {
public:
    CGameStateOver(CGame *g);
    void OnInit();
    void OnBeginState(); // 設定每次重玩所需的變數
protected:
    void OnMove();       // 移動遊戲元素
    void OnShow();       // 顯示這個狀態的遊戲畫面
private:
    int counter;         // 倒數之計數器
    CString stringHandler;
    CMovingBitmap bg;
};
}

mygame.cpp
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "mygame.h"
#include "Item.h"
#include <iostream>
#include <time.h>
#include <ctime>
using namespace std;
namespace game_framework {
// 這個class為遊戲的遊戲開頭畫面物件
// 這個class為遊戲的遊戲開頭畫面物件
CGameStateInit::CGameStateInit(CGame *g) : CGameState(g)
{
    tutro = false;
    intro_done = false;
    optionStage = -1;
}
CGameStateInit::~CGameStateInit() {
    option_sel.clear();
    menu.clear();
    option.clear();
    bin_select.clear();
    tri_select.clear();
    stages.clear();
}
void CGameStateInit::OnInit() {

```

```

ShowInitProgress(0);
// Loading Images
stringHandler.LoadBitmap();
numberHandler.LoadBitmap();
// Intro
intro.AddBitmap(INTRO_1);
intro.AddBitmap(INTRO_1);
intro.AddBitmap(INTRO_1);
intro.AddBitmap(INTRO_2);
intro.AddBitmap(INTRO_3);
intro.AddBitmap(INTRO_4);
intro.AddBitmap(INTRO_4);
intro.SetDelayCount(3);
img_tutro.LoadBitmap(TUTRO);
// Background
background1.LoadBitmap(BG_OPENING_1);
background2.LoadBitmap(BG_OPENING_2);
background3.LoadBitmap(BG_OPENING_3);
background4.LoadBitmap(BG_OPENING_4);
background5.LoadBitmap(BG_OPENING_5);
optionBG.LoadBitmap(BG_OPTION);
// Logo
logo1.LoadBitmap(LOGO_1);
logo2.LoadBitmap(LOGO_2);
logo.AddBitmap(LOGO_CHARACTER_1);
logo.AddBitmap(LOGO_CHARACTER_2);
logo.AddBitmap(LOGO_CHARACTER_3);
logo.AddBitmap(LOGO_CHARACTER_4);
logo.AddBitmap(LOGO_CHARACTER_3);
logo.AddBitmap(LOGO_CHARACTER_2);
logo.SetDelayCount(3);
// Menu
menu.push_back("START GAME");
menu.push_back("TUTORIAL");
menu.push_back("ABOUT");
menu.push_back("OPTION");
menu.push_back("EXIT");
option.push_back("FULLSCREEN");
option.push_back("STAGE SELECT");
option.push_back("BACK");
bin_select.push_back("YES");
bin_select.push_back("NO");
// stages
stages.push_back("BULE OCEAN ZONE ACT ONE");
stages.push_back("BULE OCEAN ZONE ACT TWO");
stages.push_back("EXOTIC PARADISE ACT ONE");
stages.push_back("EXOTIC PARADISE ACT TWO");
stages.push_back("TUTORIAL ACT ONE");
for (int i = 0; i < 4; i++)
    option_sel.push_back(1);
if (OPEN_AS_FULLSCREEN)
    option_sel.at(static_cast<int>(OPTION::FULLSCREEN)) = 0;
// Loading Audio
CAudio::Instance()->Load(AUDIO_CHOOSE, "sounds\\choose.wav");
CAudio::Instance()->Load(AUDIO_SELECT, "sounds\\select.wav");
CAudio::Instance()->Load(AUDIO_RETURN, "sounds\\return.wav");
CAudio::Instance()->Load(AUDIO_RING, "sounds\\ring.wav");
//CAudio::Instance()->Load(AUDIO_SPRING, "sounds\\spring.wav");
}
void CGameStateInit::OnBeginState() {
CAudio::Instance()->Load(AUDIO_TITLE, "sounds\\title.mp3");
CAudio::Instance()->Load(AUDIO_OPTIONS, "sounds\\options.mp3");
CAudio::Instance()->Load(AUDIO_CRUSADER, "sounds\\crusader.mp3");
CAudio::Instance()->Play(AUDIO_TITLE, true);
CAudio::Instance()->Stop(AUDIO_OPTIONS);
CAudio::Instance()->Stop(AUDIO_CRUSADER);
page = static_cast<int>(MENU::START_GAME);
current_select = 0;
option_select = 0;

```

```

    stageSelect = 0;
}
void CGameStateInit::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {
    const char KEY_ENTER = 0x0D; // keyboard ENTER
    const char KEY_ESC   = 0x1B; // keyboard ESC
    const char KEY_SPACE = 0x20; // keyboard SPACE
    const char KEY_LEFT  = 0x25; // keyboard左箭頭
    const char KEY_UP    = 0x26; // keyboard上箭頭
    const char KEY_RIGHT = 0x27; // keyboard右箭頭
    const char KEY_DOWN  = 0x28; // keyboard下箭頭
    if (!tutro) {
        if (nChar) tutro = true;
    }
    else {
        switch (page) {
            // START GAME
            case static_cast<int>(MENU::START_GAME) :
                if (nChar == KEY_ENTER || nChar == KEY_SPACE) {
                    CAudio::Instance()->Play(AUDIO_SELECT, false);
                    switch (current_select) {
                        case static_cast<int>(MENU::START_GAME) :
                            CGame::Instance()->SetLevel(0);
                            GotoGameState(GAME_STATE_RUN);
                            break;
                        case static_cast<int>(MENU::TUTORIAL) :
                            CGame::Instance()->SetLevel(static_cast<int>
                                (LEVELS::TUTORIAL_1));
                            GotoGameState(GAME_STATE_RUN);
                            break;
                        case static_cast<int>(MENU::EXTRAS) :
                            page = static_cast<int>(MENU::EXTRAS);
                            break;
                        case static_cast<int>(MENU::OPTION) :
                            CAudio::Instance()->Stop(AUDIO_TITLE);
                            CAudio::Instance()->Play(AUDIO_OPTIONS, true);
                            current_select = 0;
                            page = static_cast<int>(MENU::OPTION);
                            break;
                        case static_cast<int>(MENU::EXIT) :
                            Sleep(500);
                            PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
                            break;
                    }
                }
                if (nChar == KEY_UP) {
                    CAudio::Instance()->Play(AUDIO_CHOOSE, false);
                    current_select -= 1;
                }
                if (nChar == KEY_DOWN) {
                    CAudio::Instance()->Play(AUDIO_CHOOSE, false);
                    current_select += 1;
                }
                if (current_select < 0)
                    current_select = static_cast<int>(MENU::COUNT) - 1;
                else
                    current_select %= static_cast<int>(MENU::COUNT);
                break;
            // EXTRAS
            case static_cast<int>(MENU::EXTRAS) :
                if (nChar == KEY_ESC) {
                    CAudio::Instance()->Play(AUDIO_RETURN, false);
                    page = static_cast<int>(MENU::START_GAME);
                }
                break;
            // OPTION
            case static_cast<int>(MENU::OPTION) :
                switch (optionStage) {
                    case -1:
                        if (nChar == KEY_ESC) {

```

```

        CAudio::Instance()->Play(AUDIO_RETURN, false);
        CAudio::Instance()->Stop(AUDIO_OPTIONS);
        CAudio::Instance()->Play(AUDIO_TITLE, true);
        intro_done = false;
        current_select = 0;
        page = static_cast<int>(MENU::START_GAME);
    }
    if (nChar == KEY_UP) {
        CAudio::Instance()->Play(AUDIO_CHOOSE, false);
        current_select -= 1;
    }
    if (nChar == KEY_DOWN) {
        CAudio::Instance()->Play(AUDIO_CHOOSE, false);
        current_select += 1;
    }
    if (nChar == KEY_LEFT && current_select < 4) {
        CAudio::Instance()->Play(AUDIO_SELECT, false);
        option_sel.at(current_select) -= 1;
        if (option_sel.at(current_select) < 0)
            option_sel.at(current_select) =
                (current_select == 0) ? 2 : 1;
        else
            option_sel.at(current_select) %=
                (current_select == 0) ? 3 : 2;
    }
    if (nChar == KEY_RIGHT && current_select < 4) {
        CAudio::Instance()->Play(AUDIO_SELECT, false);
        option_sel.at(current_select) += 1;
        option_sel.at(current_select) %=
            (current_select == 0) ? 3 : 2;
    }
    if (nChar == KEY_ENTER || nChar == KEY_SPACE) {
        CAudio::Instance()->Play(AUDIO_SELECT, false);
        if (current_select < 1) {
            option_sel.at(current_select) += 1;
            option_sel.at(current_select) %= 2;
        }
        else {
            if (current_select ==
                static_cast<int>(OPTION::STAGE_SELECT))
                optionStage = 1;
            if (current_select ==
                static_cast<int>(OPTION::BACK)) {
                CAudio::Instance()->Stop(AUDIO_OPTIONS);
                CAudio::Instance()->Play(AUDIO_TITLE, true);
                current_select = 0;
                intro_done = false;
                page = static_cast<int>(MENU::START_GAME);
            }
        }
    }
    //
    switch (current_select) {
    case static_cast<int>(OPTION::FULLSCREEN) :
        if (CDDraw::IsFullScreen() &&
            option_sel.at(static_cast<int>(OPTION::FULLSCREEN))
            || !CDDraw::IsFullScreen()
            && !option_sel.at(static_cast<int>(OPTION::FULLSCREEN
            )))
            CDDraw::SetFullScreen(!(option_sel.at(
                static_cast<int>(OPTION::FULLSCREEN))));
        break;
    }
    if (current_select < 0)
        current_select = static_cast<int>(OPTION::COUNT) - 1;
    else
        current_select %= static_cast<int>(OPTION::COUNT);
    break;
case 0:

```

```

        if (nChar == KEY_ESC ||
            nChar == KEY_ENTER ||
            nChar == KEY_SPACE) {
            CAudio::Instance()->Play(AUDIO_RETURN, false);
            optionStage = -1;
        }
        break;
    case 1:
        if (nChar == KEY_ESC) {
            CAudio::Instance()->Play(AUDIO_RETURN, false);
            optionStage = -1;
        }
        if (nChar == KEY_UP) {
            CAudio::Instance()->Play(AUDIO_CHOOSE, false);
            stageSelect--;
            if (stageSelect < 0) stageSelect =
                static_cast<int>(LEVELS::COUNT) - 1;
        }
        if (nChar == KEY_DOWN) {
            CAudio::Instance()->Play(AUDIO_CHOOSE, false);
            stageSelect++;
            stageSelect %= static_cast<int>(LEVELS::COUNT);
        }
        if (nChar == KEY_ENTER || nChar == KEY_SPACE) {
            CAudio::Instance()->Play(AUDIO_SELECT, false);
            CGame::Instance()->SetLevel(stageSelect);
            optionStage = -1;
            GotoGameState(GAME_STATE_RUN);
        }
        break;
    case 2:
        if (nChar == KEY_ESC ||
            nChar == KEY_ENTER ||
            nChar == KEY_SPACE) {
            CAudio::Instance()->Play(AUDIO_RETURN, false);
            optionStage = -1;
        }
        break;
    default: break;
}
}
}
}
}
void CGameStateInit::OnMove() {
    if (!intro_done) {
        // Intro animate
        intro.OnMove();
    } else {
        // Background animate
        background1.SetTopLeft(0, 0);
        background2.SetTopLeft(0, (int)(SIZE_Y/3));
        background3.SetTopLeft(0, (int)(SIZE_Y/3) +
            (int)((background2.Height()-32) * DEFAULT_SCALE));
        background4.SetTopLeft(0, (int)(SIZE_Y/3) +
            (int)((background2.Height()-32 +
            background3.Height())*DEFAULT_SCALE));
        background5.SetTopLeft(0, (int)(SIZE_Y - background3.Height() -
            background4.Height() * (DEFAULT_SCALE-1)));
        // Logo
        logo.SetTopLeft((SIZE_X-logo.Width()*DEFAULT_SCALE)/2,
            SIZE_Y*2/100);
        logo1.SetTopLeft((SIZE_X-logo1.Width()*DEFAULT_SCALE)/2,
            SIZE_Y*6/100);
        logo2.SetTopLeft((SIZE_X-logo2.Width()*DEFAULT_SCALE)/2,
            SIZE_Y*46/100);
        logo.OnMove();
    }
}
void CGameStateInit::OnShow() {

```



```

if (!tutro) {
    img_tutro.SetTopLeft(0, 0);
    img_tutro.ShowBitmap();
}
else if (!intro_done) {
    intro.SetTopLeft((SIZE_X - logo.Width() * DEFAULT_SCALE) / 2,
        SIZE_Y * 2 / 100);
    intro.OnShow();
    if (intro.IsFinalBitmap()) {
        intro.Reset();
        intro_done = true;
    }
}
if (intro_done) {
    switch (page) {
        // START GAME
        case static_cast<int>(MENU::START_GAME) :
            // Background
            background1.ShowBitmap((double)SIZE_X / background1.Width());
            background2.ShowBitmap();
            background3.ShowBitmap();
            background4.ShowBitmap();
            background5.ShowBitmap();
            // Logo
            logo1.ShowBitmap();
            logo.OnShow();
            logo2.ShowBitmap();
            // Menu
            for (int i = 0; i < static_cast<int>(MENU::COUNT); i++) {
                stringHandler.SetTopLeft(SIZE_X*35/100,
                    SIZE_Y*(70+5*i)/100);
                stringHandler.SetFocus(i == current_select);
                stringHandler.ShowBitmap(menu.at(i));
            }
            break;
        // EXTRAS
        case static_cast<int>(MENU::EXTRAS) :
            // Background
            background1.ShowBitmap((double)SIZE_X / background1.Width());
            background2.ShowBitmap();
            background3.ShowBitmap();
            background4.ShowBitmap();
            background5.ShowBitmap();
            stringHandler.SetFocus(true);
            stringHandler.SetTopLeft(SIZE_X/2 -
                (2*DEFAULT_SCALE*stringHandler.GetAlphabet()->Width()),
                5*SIZE_Y/100);
            stringHandler.ShowBitmap("ABOUT");
            stringHandler.SetFocus(false);
            numberHandler.SetTopLeft(9*SIZE_X/100, 10*SIZE_Y/100);
            numberHandler.SetInteger(2020);
            numberHandler.ShowBitmap();
            stringHandler.SetTopLeft(30*SIZE_X/100, 11*SIZE_Y/100);
            stringHandler.ShowBitmap("SPRING_CSIE_NTUT");
            stringHandler.SetFocus(true);
            stringHandler.SetTopLeft(5*SIZE_X/100, 25*SIZE_Y/100);
            stringHandler.ShowBitmap("DEVELOPER");
            stringHandler.SetTopLeft(15*SIZE_X/100, 30*SIZE_Y/100);
            stringHandler.SetFocus(false);
            stringHandler.ShowBitmap("MING HSUAN HUANG");
            stringHandler.SetTopLeft(15*SIZE_X /100, 35*SIZE_Y/100);
            stringHandler.ShowBitmap("HO MING LI");
            stringHandler.SetFocus(true);
            stringHandler.SetTopLeft(5*SIZE_X/100, 45*SIZE_Y/100);
            stringHandler.ShowBitmap("INSTRUCTOR");
            stringHandler.SetFocus(false);
            stringHandler.SetTopLeft(15*SIZE_X/100, 50*SIZE_Y/100);
            stringHandler.ShowBitmap("PROF SHUO HAN CHEN");
    }
}

```

```

stringHandler.SetFocus(true);
stringHandler.SetTopLeft(5*SIZE_X/100, 60*SIZE_Y/100);
stringHandler.ShowBitmap("FRAMEWORK BY");
stringHandler.SetFocus(false);
stringHandler.SetTopLeft(15*SIZE_X/100, 65*SIZE_Y/100);
stringHandler.ShowBitmap("PROF WOEI KAE CHEN");
break;
// OPTION
case static_cast<int>(MENU::OPTION) :
    // Background
    for (int y=0; y<SIZE_Y; y+=(optionBG.Height()*DEFAULT_SCALE))
    {
        for (int x=0; x<SIZE_X;
            x+=(optionBG.Width()*DEFAULT_SCALE)) {
            optionBG.SetTopLeft(x, y);
            optionBG.ShowBitmap();
        }
    }
    // Option
    switch (optionStage) {
    case -1:
        stringHandler.SetFocus(false);
        stringHandler.SetTopLeft(SIZE_X*7/100, SIZE_Y*19/100);
        stringHandler.ShowBitmap("GRAPHICS");
        stringHandler.SetTopLeft(SIZE_X*7/100, SIZE_Y*35/100);
        stringHandler.ShowBitmap("GAME");
        for (int i=0; i<static_cast<int>(OPTION::COUNT); i++) {
            const int size = bin_select.size();
            switch (i) {
            case 0:
                stringHandler.SetTopLeft(SIZE_X*7/100,
                    SIZE_Y*(27+8*i)/100);
                stringHandler.SetFocus(i == current_select);
                stringHandler.ShowBitmap(option.at(i));
                for (int j = 0; j < size; j++) {
                    stringHandler.SetTopLeft(
                        SIZE_X*(50+15*j)/100, SIZE_Y*(27+8*i)/100);
                    stringHandler.SetFocus(j ==
                        option_sel.at(i));
                    stringHandler.ShowBitmap(bin_select.at(j));
                }
                break;
            case 1:
                stringHandler.SetTopLeft(SIZE_X*7/100,
                    SIZE_Y*(37+8*i)/100);
                stringHandler.SetFocus(i == current_select);
                stringHandler.ShowBitmap(option.at(i));
                break;
            case 2:
                stringHandler.SetTopLeft(SIZE_X*5/100,
                    SIZE_Y*90/100);
                stringHandler.SetFocus(i == current_select);
                stringHandler.ShowBitmap(option.at(i));
                break;
            }
        }
        break;
    case 1:
        stringHandler.SetFocus(false);
        for (int i=0; i<static_cast<int>(LEVELS::COUNT); i++) {
            stringHandler.SetFocus(i == stageSelect);
            stringHandler.SetTopLeft(25, 50+SIZE_Y*(8*i)/100);
            stringHandler.ShowBitmap(stages.at(i));
        }
        break;
    default: break;
    }
}
}

```

```

}
////////////////////////////////////////////////////
// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡
////////////////////////////////////////////////////
CGameStateRun::CGameStateRun(CGame *g) : CGameState(g) {
    current_level = CGame::Instance()->GetLevel();
}
CGameStateRun::~CGameStateRun() {
    if (level != nullptr) {
        level->clean();
        delete level;
    }
}
void CGameStateRun::OnInit() {
    stringHandler.LoadBitmap();
    Score.LoadBitmap(GUI_SCORE);
    Time.LoadBitmap(GUI_TIME);
    Rings.LoadBitmap(GUI_RING);
    colon.LoadBitmap(T_TIME_COLON);
    score.LoadBitmap();
    t_min.LoadBitmap();
    t_sec.LoadBitmap();
    rings.LoadBitmap();
    current_actor = 0;
    game_time = 0;
    score.SetInteger(0);
    t_min.SetInteger(0);
    t_sec.SetInteger(0);
    rings.SetInteger(0);
    debugMODE = false;
    level = nullptr;
}
void CGameStateRun::OnBeginState()
{
    current_level = CGame::Instance()->GetLevel();
    game_time = 0;
    t_min.SetInteger(0);
    t_sec.SetInteger(0);
    rings.SetInteger(0);
    level = new Level(current_level);
    level->OnInit(current_level);
}
void CGameStateRun::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    const char KEY_ESC   = 0x1B; // keyboard ESC
    const char KEY_LEFT  = 0x25; // keyboard左箭頭
    const char KEY_UP    = 0x26; // keyboard上箭頭
    const char KEY_RIGHT = 0x27; // keyboard右箭頭
    const char KEY_DOWN  = 0x28; // keyboard下箭頭
    const char KEY_SPACE = 0x20; // keyboard SPACE
    const char KEY_Z     = 0x5A; // keyboard z
    const char KEY_Q     = 0x51;

    if (nChar == KEY_Q) {
        debugMODE = !debugMODE;
        level->SetDebug(debugMODE);
    }
    /*
    if (nChar == KEY_Z) {
        ++current_actor %= 3;
        level->SetCurrentActor(current_actor);
    }
    */
    if (nChar == KEY_LEFT)
        level->CurrentActor()->SetMoveLeft(true);
    if (nChar == KEY_RIGHT)
        level->CurrentActor()->SetMoveRight(true);
    if (nChar == KEY_UP)
        level->CurrentActor()->SetIsLookingUp(true);
}

```

```

    if (nChar == KEY_DOWN)
        level->CurrentActor()->SetIsLookingDown(true);
    if (!debugMODE && nChar == KEY_SPACE)
        level->CurrentActor()->SetIsJumping(true);
    if (debugMODE && nChar == KEY_UP)
        level->CurrentActor()->SetMoveUp(true);
    if (debugMODE && nChar == KEY_DOWN)
        level->CurrentActor()->SetMoveDown(true);
    if (nChar == KEY_ESC) {
        CAudio::Instance()->Play(AUDIO_RETURN, false);
        CGameState::GotoGameState(GAME_STATE_INIT);
        level->clean();
    }
}
void CGameStateRun::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    const char KEY_LEFT = 0x25; // keyboard左箭頭
    const char KEY_UP = 0x26; // keyboard上箭頭
    const char KEY_RIGHT = 0x27; // keyboard右箭頭
    const char KEY_DOWN = 0x28; // keyboard下箭頭
    const char KEY_SPACE = 0x20; // keyboard SPACE
    if (nChar == KEY_LEFT)
        level->CurrentActor()->SetMoveLeft(false);
    if (nChar == KEY_RIGHT)
        level->CurrentActor()->SetMoveRight(false);
    if (nChar == KEY_UP)
        level->CurrentActor()->SetIsLookingUp(false);
    if (nChar == KEY_DOWN)
        level->CurrentActor()->SetIsLookingDown(false);
    if (!debugMODE && nChar == KEY_SPACE)
        level->CurrentActor()->SetIsJumping(false);
    if (debugMODE && nChar == KEY_DOWN)
        level->CurrentActor()->SetMoveDown(false);
    if (debugMODE && nChar == KEY_UP)
        level->CurrentActor()->SetMoveUp(false);
}
void CGameStateRun::OnMove()
{
    if (CGame::Instance()->GetFinish() || CGame::Instance()->GetDead())
        GotoGameState(GAME_STATE_OVER);
    level->OnMove();
    game_time++;
    int t_time = game_time / GAME_CYCLE_TIME;
    int mins = t_time / 60;
    int sec = t_time % 60;
    t_min.SetInteger(mins);
    t_sec.SetInteger(sec);
    score.SetInteger(level->getScore());
    rings.SetInteger(level->getRings());
}
void CGameStateRun::OnShow() {
    level->OnShow();
    Score.SetTopLeft(32, 20);
    score.SetTopLeft(64, 20);
    Time.SetTopLeft(32, 50);
    t_min.SetTopLeft(50, 50);
    colon.SetTopLeft(137, 56);
    t_sec.SetTopLeft(72, 50);
    Rings.SetTopLeft(32, 80);
    rings.SetTopLeft(64, 80);
    Score.ShowBitmap();
    Time.ShowBitmap();
    Rings.ShowBitmap();
    colon.ShowBitmap();
    score.ShowBitmap();
    t_min.ShowBitmap();
    t_sec.ShowBitmap();
    rings.ShowBitmap();
}

```

```

////////////////////////////////////
// 這個class為遊戲的結束狀態(Game Over)
////////////////////////////////////
CGameStateOver::CGameStateOver(CGame* g) : CGameState(g) { }
void CGameStateOver::OnInit() {
    stringHandler.LoadBitmap();
    bg.LoadBitmap(BG);
    CAudio::Instance()->Stop(AUDIO_CRUSADER);
}
void CGameStateOver::OnBeginState() {
    counter = 30 * 3;
}
void CGameStateOver::OnMove() {
    if (--counter < 0) {
        GotoGameState(GAME_STATE_INIT);
        CGame::Instance()->SetDead(false);
        CGame::Instance()->SetFinish(false);
    }
}
void CGameStateOver::OnShow() {
    bg.SetTopLeft(0, 0);
    bg.ShowBitmap();
    stringHandler.SetFocus(true);
    if (CGame::Instance()->GetDead()) {
        stringHandler.SetTopLeft(SIZE_X/2-4*DEFAULT_SCALE*
            (stringHandler.GetAlphabet()->Width()), SIZE_Y/2-
            stringHandler.GetAlphabet()->Height());
        stringHandler.ShowBitmap("YOU DEAD");
    }

    if (CGame::Instance()->GetFinish()) {
        stringHandler.SetTopLeft(SIZE_X/2- 4*DEFAULT_SCALE*
            (stringHandler.GetAlphabet()->Width()), SIZE_Y/2-
            stringHandler.GetAlphabet()->Height());
        stringHandler.ShowBitmap("YOU WIN");
    }
}
}
}

```