組員：

資工二　B95902101　曾冠諭

資工二　B95902055　李東耿

資工二　B95902017　翁健庭

資工二　B95902009　徐銘遠

目錄

I.Prefrace：

在童年時期，放學後或假日的閒暇時間常常會呼朋引伴一起玩紙上大富翁遊戲，一玩就是整個下午。大家玩的時候都非常興奮投入，聚精會神的注視著棋盤，每一次的擲骰彷彿都牽動著喜怒哀樂。沒多久後，許多遊戲廠商，諸如大宇也推出了許多類似大富翁的遊戲，例如萬種棋盤、非洲尋寶、幸福人、奮鬥人生及大富翁世界版等，甚至還有加入殭屍怪談風格的大富翁遊戲。玩家的熱烈迴響，也使得國產遊戲除了武俠角色扮演類型外，最受歡迎的就是大富翁類型的遊戲。但其中最受歡迎的還是大宇資訊推出的大富翁系列。大宇資訊的《大富翁 》系列自一代推出後十幾年來始終如一，不但保留傳統大富翁的遊戲風格，尚不斷地求新求變，每一代推出的大富翁都能帶給玩家不同的驚喜及體驗。也難怪本系列能在眾多同類型遊戲中脫穎而出，成爲銷售最佳、支持度最高的《大富翁》系列遊戲。

承襲至 rich man1~10 代，以及 rich man online，這款專爲 GBA 平台設計的【台大富翁】，標榜著佔用小容量記憶體，堅持小遊戲的迷你有趣，把大富翁的精神用小平台詮釋的唯妙唯俏。

1. 遊戲進入畫面以輕鬆活潑降落傘動畫做起頭，那塊匾額式遊戲名稱台大富翁，則是依據之前的大富翁 logo，並將設計理念從傳承中做個統合，設計而成，整個標題以金黃燙字爲底，將錢財滾滾的喜悅充分表達。
2. 遊戲功能上，機會、命運、骰子、破產及坐牢，擲骰子、走方格、買土地、蓋房子已經成爲大富翁的基本架構，在本遊戲中一開始你有四位角色可以做選擇，每人身上會有預定金額，銀行內會有預訂存款，前進模式跟傳統的擲骰子模式雷同，由動畫骰子的點數決定前進步數，若來到空地則可以買地蓋房子，若別的玩家經過則可以收取其過路費，若走到銀行，系統會詢問你是存款或提款，比較特別的是，機會跟命運會有些觸發事件，而這些觸發事件可能讓你進監獄囚禁幾回合或者受傷進醫院，當然有是有可能得到可觀利益的。
3. 結尾畫面，Game Over 圖利用效果上的反差，包準讓玩家瞠目結舌，呆立當場，最終的感謝名單，則以電影的方式呈現，文字的浮動搭配深有涵義 background，將韻味完整呈現，最後的 the end 在夕陽下隱約透出，發人深省。

II. Research Motivation：

地產大亨（又稱大富翁）爲全球最受歡迎的遊戲之一。自1904年在美國推出以來，大富翁紙盤遊戲不斷轉變，後來，國內大宇資訊公司甚至將其軟體化，製成能在電腦遊玩的軟體，還記得，當時第一次接觸到這款由紙盤大富翁移植到電腦的遊戲軟體時，內心是多麼的開心振奮！因此，當我們小組在討論究竟這學期的期末報告要以什麼作爲研究主題時，很快的就想到了這款飽受眾人喜愛的大富翁遊戲，希望能藉由自己的手，作出屬於自己的大富翁，將當初接觸到大富翁遊戲軟體那種歡喜振奮的心情傳達出來。

III. Introduction to the game：

Step1.按下 start 鍵開始進行大富翁遊戲

出現右圖畫面，按下 E nter 鍵
進入選擇人物畫面

S tep2.進行選擇角色

1.按下 A 及選擇該腳色加入遊戲
（至少要選二人）
2.按下 Ebter 即可進入遊戲畫面

Step3.

- **擲骰子**

按下 A 直接擲骰子決定移動步數

- **使用道具（如右圖）**

1. 按下 select 鍵進入選單
2. 選擇 item 按下 A
3. 進入道具選單
4. 選擇欲用的道具及可

- **位置、狀態查詢(如右圖)**

1. 按下 select 鍵進入選單
2. 選擇 my status 即可看目前狀態
   選擇 place 即可看位置

- **機會命運**

遊戲設有如紙盤大富翁的機會命運踩到後會產生不同之效果

- **銀行**

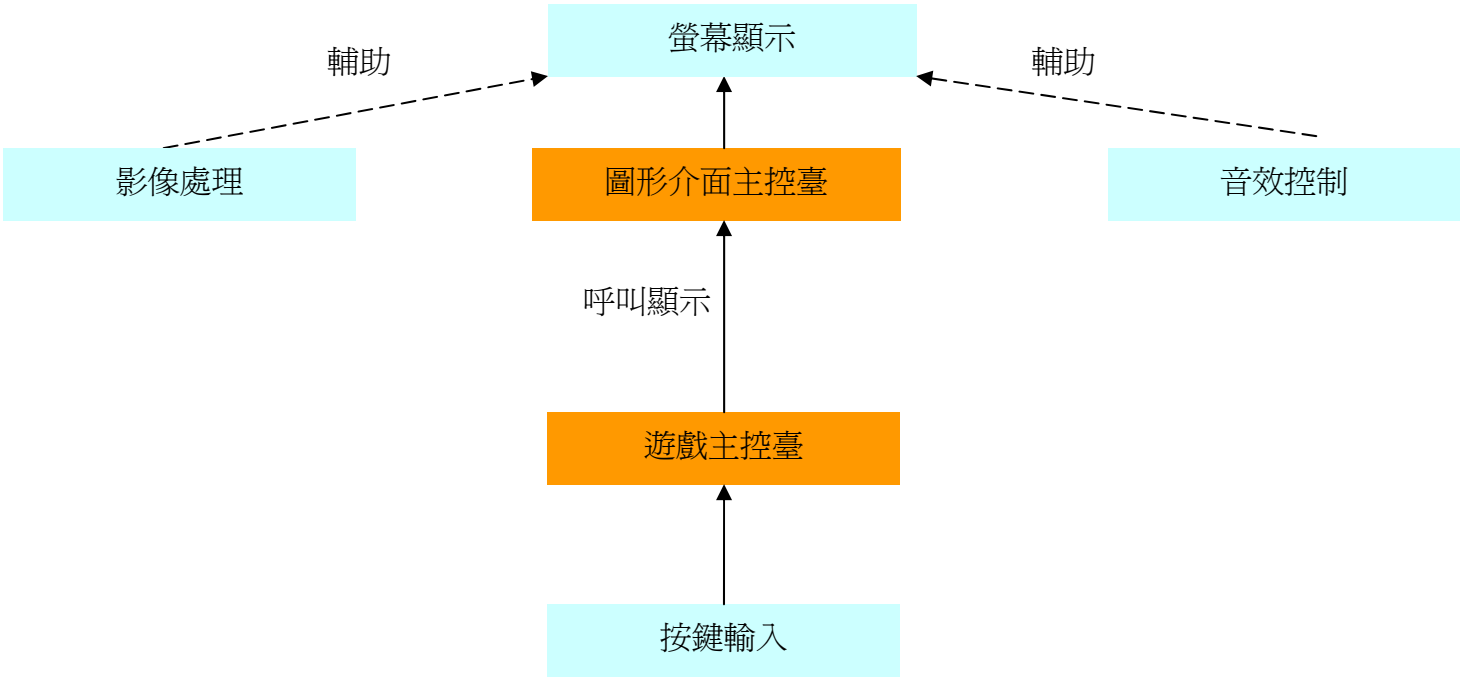
同大宇大富翁系列之銀行，經過可以存錢或領錢

III. Implementation：

設計架構：

在 code 的部分，爲了更貼近組合語言的架構，所以我們這組選擇使用 Tonc 的 library 來設計遊戲，雖然採用 C 當作主要的架構，但是不同於 HAM，Tonc Library 在 GBA programming 已經非常低階，所以已經非常貼近組合語言的架構。接著描述我們整個程式的設計架構：

我們這次台大富翁的設計架構主要是分成兩部份進行，一部份開發**大富翁遊戲主控台**的部份，一部分開發**圖形介面**的部份，所以有一組人先用 C 語言寫一個文字介面的大富翁，包含擲骰子、走路、判斷機會命運，買地賣地之類大富翁主控的功能，而另外一組人則是開發圖形介面的部份，兩部份最後就利用主控台呼叫圖形介面所開發出來的 function，例如說主控台擲了六點，要螢幕上的人物走六步，可能從(100,300)走到(100, 370)，這時候主控臺就呼叫 move_to_pos(&player, x, y)這個函式，而負責圖形界面的函式就會讓螢幕上的人物從(100, 300)移動到(100,370)，而且因爲大富翁是屬於**回合制**的遊戲，大部份按鍵不需要及時的回應，所以可以分成這兩部份進行。另外，**音效部分**也同步進行，寫出函式讓主控台去呼叫，而 GBA 的圖形界面上，需要大量的**影像處理**，例如說把圖片縮小、轉 16 色、去背之類的。而開場畫面的**動畫**，因爲和遊戲沒有什麼直接的關係，所以也可以分開進行。



遊戲主控臺：

要用 C 的程式寫出文字版的大富翁程式，在設好一切 struct 後，主要是根據使用者輸入的資料，去更改人物和土地的各項參數。等到 C 的部分寫好了，再把本來是輸出到 stdout 的文字改成不斷印在 GBA 的螢幕上(需要注意，僅有文字部分需要不斷輸出)，每件本來在使用者輸入按鍵後會執行的事，改成在使用者按下 GBA 中的按鍵再去作判斷與執行(遊戲中會出現[]來表示現在選擇到的選項，需要作一些判斷來決定[]的位置，在此程式中通常是以 status 這個變數做判斷)。所以首先，把主要人物，土地，道具，機會和命運用 struct 來表示，

第一步先建立 XXXdefine.h 檔，把像是人物數量，土地數量，機會命運卡數等在遊戲過程中不會更動的常數在此些檔中定義好，以 mapdefine.h 爲例，主要是 define 土地總數量(PLACENUM)，每個土地可以建的房子總數(MAXHOUSE)，其他的參數則是方便在過程中定義 type 時可以用比較口語話的方式去寫(比方說寫 map[10].type = norma，代表這是一個正常的地)。

第二步建立 XXXstructdef.h 檔，include 上面的 XXXdefine.h，將所有遊戲中的物件會有的參數定義在這裡面，然後用一個陣列把這些物件都存進去，以後就直接讀取這些陣列來讀取物件。

第三步建立 XXXinitial.h 檔，include 上面的 XXXstructdef.h，手動設定所有土地，人物的資料，比方說像土地和人物名稱、地價，初始金錢等全都在這些檔中設定，以後像是要測試破產、遊戲結束等要在遊戲中進行一陣子才能達到的事件時，只要把所有人金錢都初始化負數就可以達到。

主要的 main.c，僅處理人物選單、畫面移動和遊戲進行過程等問題，其他像是走路、使用道具、破產等對個別人物執行的動作，都用其他的.h 檔寫出 function 後給 main.c 來 inlcude 使用：

1.Selectplayername.h：

處理初始選擇的人物，由於此遊戲要兩個人以上去玩，所以要擋下使用者選擇人物不到兩個的情況。選好人物後，把人物的資料移到上面定義的陣列裡，把不要選的人物隱藏起來，就可以開始遊戲了。在選擇遊戲人物的過程中，定一個變數去紀錄此 while(1)迴圈跑了幾次，然後依此變數去當作 srand()的 input，以產生亂數。

2.run.h：

處理人物移動的動作。給予現在要移動的人物、移動的步數，然後移動此人物到他該移動的地點。要特別去寫經過銀行時的動作，包含如何提、存款，發放利息(在離開的時候根據現在存款發放 5%的利息)。

3.bulidorpay.h：

處理人物到達定位後要作的事情。共分三種情況：走到無主地、走到自己的地、走到其他人的地。走到無主地就給予買地選項，依人物現金決定是否買地；走到自己的地則視房子最大數量詢問是否要加蓋房子，再依現金決定是否加蓋；走到別人的地則算出地價(連鎖反應是否在同類型的地上有同樣的主人，若有的話還要再增加過路費)，扣除現金(現金不夠則扣存款)。

4.bankrupt.h：

處理人物破產的情況，判斷條件爲現金加上存款小於零。如果某一個人物破產了，則把此人物設爲隱藏，他所擁有的土地全變爲無主地(保留房子)。

5.chancecard.h：

走到機會會發生的事件。這些事件要根據 initial 出的事件去更改，在此版本中，可能發生的事件爲：撿到 500 元、被狗咬而送醫、轉向、發出 1000 元紅包、所有的其他人均暫停一次。

6.destinycard.h：

走到命運會發生的事件。和機會一樣，會根據 initial 更改發生的事件，此版本可能發生的事件爲：打架送警局、違反交通規則罰款 1500、銀行發放所有人利息、打工(休息一天)，得到 2000 元、直接移到銀行。

7.item.h：

如果人物選擇使用道具時要進行的事件，包含使用遙控骰子、飛彈。遙控骰子會讓使用者選擇此回合要骰出的點數，飛彈則是攻擊自己以外的任何一個人，被攻擊者住院三天。

影像處理：

我們的處理主要分爲 background 跟 sprite，主要爲 tile 的處理，tile 是以 8x8 的 pixel 爲基本單位，在 sprite 的格式基本上介於 8x8，64x64pixel 之間，而 backgroud 則是有四種 size 介於 128x128 至 1024x1024 之間，值得一提的是 sprite 的 shape 又分爲 sqare，tall 跟 wide。Per tile size：基本上 4bpp 搭配 16 色，若 8bpp 搭配 256 色，以上這些參數在 Object 的 attribute 的 attr0~attr3 的設定上都要注意，這部份的參數可以參考 Tonc Manual，在 Load graph 方面，GBA 因爲 VRAM 不是很大，如果讀入圖形太多，顯示上會有圖塊不完整的問題。

在圖形設計方面，我們會用到套索，跟圖層分割，合成，淡化修圖，調整 pixel 等功能，這方面我們借助 photoshopCS3 跟 PhotoImpact、小畫家。

至於 Tile Set 的部份，在製作上我們一開始在 Mappy 上尋求協助，不過發現其無法將一般 bmp 的圖切成 tile set，另一個 jar 的軟體，使用上頗佔記憶體，也被捨棄，最後找到一個叫 Tile Studio，它可將任意格式的圖轉進切成 tile，Tile 在 size 上的設定跟 pixel 上的調整就變的相當方便，我們的大地圖即是借助其功能設計完成。寫 GBA 一定會用的是轉圖的程式，Tonc 作者推薦的是 Usenti，它可以把

我們 sprite 因爲數量不少，故在 sprite 的 attr0~attr3 的存取上有用到 stack，這個在待會會說明，值得注意的是在 SBB(Screen Base Block) 跟 CBB(Char Base Block)記憶體的存取常常會有重疊導致圖被破壞的情形。

Related Code：

Sprite：讀入一個 Sprite 的方法

Code:

```
//建立一個 buffer 陣列來暫存 OAM(Object Attribute Memory)
OBJ_ATTR obj_buffer[128];
//初始化此 buffer
oam_init(obj_buffer, 128);
//將圖形所轉出來的圖形陣列及調色盤複製到 tile memory
memcpy(&tile_mem[4][0], floorTiles,floorTilesLen);
memcpy(pal_obj_mem,floorPal,floorPalLen);
//設定 Display Control 的 Register 成顯示 Sprite 的 mode
REG_DISPCNT= DCNT_MODE0| DCNT_OBJ | DCNT_OBJ_1D;
//建立一個 Object attribute
obj_set_attr(&(obj_buffer[0].attr),ATTR0_SQUARE | ATTR0_4BPP,
ATTR1_SIZE_16x16, ATTR2_PALBANK(pb) | tid);
//設定一個 Object 的座標(x, y)
obj_set_pos(&(obj_buffer[0].attr), x, y);
//將建立的 Object attribute 放到 OAM
oam_copy(oam_mem, &obj_buffer[0], 1);
```

Background：建立一個 Background 的方法

Code:

```
//將轉出來的圖形陣列、調色盤和 tilemap 存到 tile memory 去
memcpy(pal_bg_mem, graphPal,graph PalLen);
memcpy(&tile_mem[0][0], graphTiles, graphTilesLen);
memcpy(&se_mem[30][0],game_over_3_Map,game_over_3_MapLen);
//設定 Rigister 的參數，這裡是把 tilesets 放到 CBB 0, 把 tilemap 放
//到 SBB 30，background 是 4bit 顏色，大小是 64*32
REG_BG0CNT= BG_CBB(0) | BG_SBB(30) | BG_4BPP |
BG_REG_64x32;
//並將 Display Control Register 設成以 Mode 0 顯示 BG0
REG_DISPCNT= DCNT_MODE0 | DCNT_BG0;
//設定螢幕在 tilemap 上的座標
REG_BG_OFS[0].x=x;
REG_BG_OFS[0].y=y;
```

音效控制：

在GBA上欲控制聲音，必須一次控制許多Register，REG_SOUNDCNT_H、REG_TMxCNT、REG_TMxD、REG_DMAxSAD、REG_DMAxDAD、REG_DMAxCNT及中斷停止用的REG_IE、REG_IF，且TONC　Library並無很好的音樂播放函式，因此要有效控制聲音極不容易。我們採用網路上他人以寫好的source code，將其理解並進行更改，大體上code的內容便是REG_TM0D、REG_TMxCNT控制timer再將來源檔案的address傳送到REG_DMAxSAD、REG_DMA1DAD記下來原檔案要傳送到的位置，最後利用REG_DMA1CNT　控制Direct Sound channels來播放音樂。另外，因播放停止點不甚了解如何控制，因此以暴力的方式利用Timer Registor將播放完的音樂強迫停止。

Related code：

Step1:先利用 TektronicWave 軟體將欲播放的 wav 音樂轉成 sound.c 的檔案，內容皆爲 PCMSOUND 的 structure。

```
typedef struct _pcmsound{
        const char * pName;                 //此爲音樂轉出來的陣列
        const unsigned int nSamplingRate;   //此爲音樂的 frequency
        const unsigned long nLength;        //此爲音樂的播放長度
    } PCMSOUND,* PPCMSOUND;
```

Step2:利用 play_sfx 函式播放音樂：

```
REG_TM0D、REG_TMxCNT        //控制 timer
REG_DMAxSAD                 //來源檔案的 address
REG_DMA1DAD                 //來源檔案要傳送到的位置
REG_DMA1CNT                 //控制 Direct Sound channels 來播放音樂
void play_sfx(PCMSOUND b){
// make sure Timer 0 is off
    REG_TM0CNT = 0;
// make sure DMA channel 1 is turned off
    REG_DMA1CNT = 0;
// make sure the FIFO is reset
    REG_SOUNDCNT_H |= 0x0800;    // just set the reset bit and leave the other ones
                                    alone
// start the timer using the appropriate frequency
    REG_TM0D     = 0xFBE8;
    REG_TM0CNT = 0x0080;
// start the DMA transfer on channel 1
```

```
    REG_DMA1SAD = (u32)b.pName;
    REG_DMA1DAD = 0x040000A0;
    REG_DMA1CNT = 0x80000000 | 0x30000000 | 0x04000000 | 0x02000000;
}
```

開頭動畫：

開頭動畫是利用一張比較大的圖，把它一個一個的畫面，然後慢慢播放所製造出來的，而降落傘畫面則是利用 Sprite 的方法讓他一直顯示出來，台大富翁的 Mark 爲什麼會慢慢彈最後停下來，是利用類似非彈性碰撞的計算方法，讓它停下來的。

圖形介面主控台：

使用 GBA 寫圖形介面有一個最大的問題就是是記憶體有限，所以怎麼管理一個記憶體，是一個很重要的問題。在 GBA 上 VRAM 不大，只有 96kB 可以使用，而大富翁又需要使用到大地圖，例如一張 512x512 的圖，要把整個它放進去 VRAM 而沒有使用到重複的 tile 的話，就快要把整個 VRAM 塞滿了。而我們是使用 GBA 的 Mode 0 模式，有四個背景可以使用，但是 GBA 支援的 map 最大的大小僅限於 512*512，如果要更大的話，就要自己對記憶體做管理，所以爲了讓我們大富翁也可以塞下 1024*1024 的地圖，就要採用另外的方法。我們採用的方法是，每當要在螢幕上要移動的地圖時候，就更新 Screen Base Block 的某一行或某一列，這樣一來，要放多大的地圖都可以了，但前提是 tiles 的數量不能太多。

另外，如果每個 Sprite 都指定給他一個 OAM 裡面的一個 OBJ_ID，那這樣頂多只能有 128 個東西(因爲 OBJ_ID 只有 128 個)，但有些東西可以現在暫時還不會用到，所以爲了避免這種佔用 OAM 記憶體的情形，所以我們利用宣告一個 obj_buffer 當作一個 stack，有需要顯示在螢幕上的 Sprite 就把它 push 進去，這樣就可以避免沒顯示在螢幕上的 Sprite 卻會佔用 OBJ_ID 的情形。會想到用這個方法是因爲我們去研究 GBA 的遊戲，發現他們也都有利用這個方式再安排記憶體，所以我們也採用同樣的方式進行。

V. What we learn：

這次的 project，我們採取先分頭進行各自的工作，最後再合併成一個完整程式的策略。在過程中也遇到許多問題，比方說要同時執行程式，又要在螢幕上顯示出文字，而 code 間協調也不算良好，雖然沒有宣告到同一變數，但卻有思考方向不同造成寫出的程式邏輯不同，造成合併上的困難的問題。

要寫出 C 版本的文字大富翁實際上並不困難，我們遇到最大的瓶頸，是互相把彼此作出的成果合併時，有不知道對方的 code 到底寫了些什麼的問題。在這裡，養成爲自己的程式下註解的習慣很重要，不但可以看自己的程式到底寫了什麼東西，又可以讓其他人容易了解。另一個遇到的麻煩是在 GBA 上印出文字。

因爲在 C 中要印出文字是很簡單的事，可是在 GBA 上，必須不斷清畫面，又要把要印的字放在 while 迴圈裡，結果造成在把 C 轉成 GBA 的 code 時，轉換花上非常多的時間，沒有效率。

在寫 code 的過程中，定義好每個遊戲中可能會因地圖和人口而變的參數是很重要的。我們在初期測試時，是先用 20 個地的地圖在玩遊戲，但是再後面要作出成果時，要改成用 44 塊地作出結果，此時只要改動初始化的地圖，和重新定好「地圖數量」爲 44 就可以了，不需要對內部 code 作其他修動。

記憶體遭受的問題也很多，GBA 所允許的記憶體並沒有我們現在通常寫的語言多，用普通想法去畫圖再載入，常會碰到記憶體不足的問題。不管是動畫或著把 512 X 512 的地圖載到 GBA 中都不是一件簡單的事。

還有是取亂數的問題，雖然可以呼叫 C 程式中的 rand 函數，可是卻沒有一個好的亂數種子丟進 srand()，我們採用的解決方法爲去計算一個 while(1)迴圈會跑多少次，然後以此當作亂數種子丟入迴圈。

在這次的 project 中，我們學到了如何使團隊合作更有效率，比方說事情把要定義的函數都先講好，才不會在結合時發生重複 define 的錯誤，還有每個人都要徹底了解其他的人主要工作爲什麼，要作結合時才能直接去找應對的人。不過，爲了結合，我們還是做了很多比較多餘的事情，比方說爲了應對初期的協商不良，必須建一個多餘的 function 讓兩邊的資料可以互相轉換，雖然跑起來並不會花太多時間，但是始終是多餘的東西。

寫起 GBA code 實在不如寫高階語言輕鬆，常要注意它在內部所給的 register 的哪個地方寫哪個參數，就算是引用他內部給的 function，常常也是有很多限制。要寫出一個好的組語程式實在不簡單，多人合作時更是有賴大家的互相溝通。

VI. Work Distribution：

ProjectLeader	：曾冠諭
Programmer	：徐銘遠
Sound	：翁健庭
Art	：李東耿
SpecialThanks	：許晉東

VII. Resource:

http://www.gamedev.net/reference/articles/article1823.asp

http://belogic.com/gba/

http://www.coranac.com/tonc/text/

http://belogic.com/gba/

Software：

TektronicWave	處理音效程式
PhotoShop/PhotoImpact	影像處理程式
Usenti	影像轉 gba 可用陣列程式
Tile Studio	處理 tile 的程式
Mappy	處理 tile 的程式
DevKitPro	Compiler