計算機組織與組合語言

期末專題

GBA programing in HAM

Strike!!

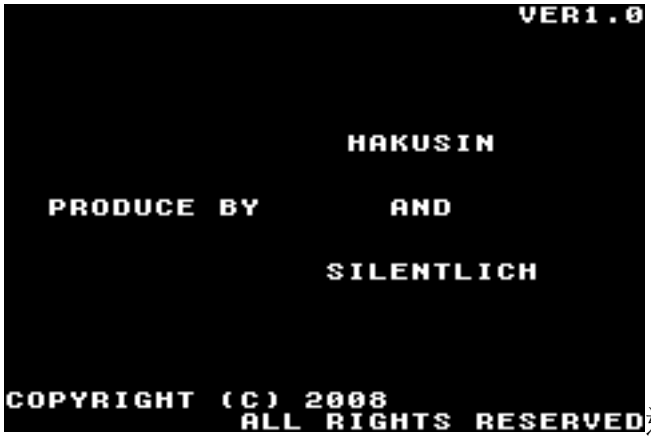
Programmer:

B95902048 資訊系 林仲丘

B95902085 資訊系 王舜玄

目錄

遊戲介紹：



這是遊戲開始的版權頁，本遊戲由hakusin與silentlich合力完成，謝謝玩家撥冗玩它，您的意見將是我們改進的助力。

這是遊戲的介紹頁，

本遊戲是射擊遊戲，移動轉心去瞄準動態的敵人，按下A鍵可以攻擊並消滅敵人，在子彈用完的時候按下B鍵可以補充彈匣，注意玩家只有六個彈匣，當彈匣用完就是遊戲該結束了。


這是在螢幕下方，顯示的玩家生命值，遭受一次攻擊會減少一個，當愛心都沒了以後遊戲就結束了，每個8x8bit。


這是玩家彈藥數，按A會減少、空了後按B會補充，每個16x8bit。

 這是作者自己畫的敵人圖案，

根據敵人位置與狀態的不同會讀取不同的片段，每個16x32bit。分別是：
向右伸腳、向右收腳、向左伸腳、向左收腳、向右開火、被消滅、向左開火。

 這是準心圖案，根據不同方向、是否開火讀取不同片段，每個32x32bit。

這是遊戲的結束畫面，

玩家可以按下start重新開始遊戲。

 這是用來計算分數的圖片，分數會和結束背景一起顯示出來，每個數字8x8bit。

參數介紹：

```
#include <mygba.h>
```
我們在這裡引用ham的內建函式庫。

以下用到生成圖片的技巧。

```
// gfx2gba -fsrc -m -pb.pal -t8 water.bmp bg1.bmp bg2.bmp gameover.bmp
water_atked.bmp
#include "gfx/bg.pal.c"
#include "gfx/bg.raw.c"
#include "gfx/bg1.raw.c"
#include "gfx/bg2.raw.c"
#include "gfx/gameover.raw.c"
#include "gfx/water.raw.c"
#include "gfx/water_atked.raw.c"
```

以上引用由指令gfx2gba建立轉換的背景圖形陣列檔案與調色盤(.pal)；

遊戲背景檔名water是因為一開始是用藍色背景。

```
// gfx2gba -D -fsrc -pobject.pal -t8 red_aiming_anim.bmp heart.bmp
bbullet.bmp num.bmp ene.bmp
#include "gfx/object.pal.c"
#include "gfx/heart.raw.c"
#include "gfx/bullet.raw.c"
#include "gfx/num.raw.c"
#include "gfx/red_aiming_anim.raw.c"
#include "gfx/ene.raw.c"
```

以上引用由指令gfx2gba建立轉換的物件圖形陣列檔案與調色盤(.pal)。

```
//定義常數
#define ANIM_UP 0
#define ANIM_UPRIGHT 1
#define ANIM_RIGHT 2
#define ANIM_DOWNRIGHT 3
#define ANIM_DOWN 4
#define ANIM_DOWNLEFT 5
#define ANIM_LEFT 6
#define ANIM_UPLEFT 7
```

以上定義準心的方向以數字0~7來表示，將會在query_buttom函式用到。

```
#define AIMING_CONT 40
```
定義準心延遲時間為40格(註：每格六十分之一秒)

```
#define BULLET_CONT 6
```
定義每匣子彈數為6個。

```
#define ENE_NUM 10
```
定義一次循環出現十個敵人。

```
#define ENE_TIME 100
```
定義敵人出現間格為100格。

```
#define ENE_ATK_TIME 240
```
定義敵人攻擊間格為240格(4秒)。

```
#define MIXER_FREQ 26757      定義音效函式的參數。

// 全域變數

u16 dir_aiming = ANIM_RIGHT;準心的方向狀態；預設為向右的狀態。
u8 aiming;                    準心物件的指標。
u8 heart[6];                  指向每個生命值物件的指標。
u8 bbullet[6];                指向每個子彈物件的指標。
u8 ene[ENE_NUM];              指向每個敵人物件的指標。
u8 aiming_x = 110 ;           預設準心的x座標。
u8 ene_x[ENE_NUM];            存放每個敵人物件的x座標的陣列（與下一個合作）。
u8 ene_pre_x[ENE_NUM];        存放每個敵人物件的前一個x座標的陣列。
int ene_count_x[ENE_NUM];     敵人的圖片計量，用來實作視覺延遲。
u8 aiming_y = 50 ;            預設準心的y座標。
u8 ene_y[ENE_NUM];            存放每個敵人物件的y座標的陣列。
u8 visidual_count ;           準心發射轉換圖片的延遲計量。
u8 a_press_ava = 0;           每次按攻擊鍵時事否為有效攻擊。
u8 base_bullet = 6 ;          玩家的彈藥數。
u8 base_box = 6 ;             玩家的彈匣數。

int i;                        讓各處迴圈使用的計量數。
int number_pic[11];           存放數字圖片的指標。
int blood_map = -1;           顯示遭受攻擊改變背景的計數。
int base_hp = 6 ;             玩家生命值。
int ene_exist[ENE_NUM] ={0};第幾號敵人是否存在。
int a_press_count = AIMING_CONT; 強制發射時間相隔
int k = 1;                    時間循環計量。
int base_score=0;             玩家得分計算。
extern const WaveData _binary_bgm_raw_start;
extern const WaveData _binary_sound1_raw_start;
extern const WaveData _binary_sound2_raw_start;
extern const WaveData _binary_sound3_raw_start;
sample_info *mysample[4];
```

主函式介紹：

```
int main() {
    ham_Init();                    初始化mygba.h函式。
    ham_InitText( 2 );             初始化文字模式。
    ham_InitMixer(MIXER_FREQ);     初始化音效模式

    mysample[0] =  ham_InitSample((u8*)_binary_bgm_raw_start.data,
                   _binary_bgm_raw_start.size,
                   _binary_bgm_raw_start.freq>>10);          //bgm
    mysample[1] =  ham_InitSample((u8*)_binary_sound1_raw_start.data,
                   _binary_sound1_raw_start.size,
                   _binary_sound1_raw_start.freq>>10);       //射擊音效
    mysample[2] =  ham_InitSample((u8*)_binary_sound2_raw_start.data,
                   _binary_sound2_raw_start.size,
                   _binary_sound2_raw_start.freq>>10);       //被打的音效
    mysample[3] =  ham_InitSample((u8*)_binary_sound3_raw_start.data,
                   _binary_sound3_raw_start.size,
                   _binary_sound3_raw_start.freq>>10);       //gamever
```

以上定義四個指向音效文件的指標，分別是背景音樂（ham內建），玩家射擊音效、敵人射擊音效、遊戲結束的音效。

```
    ham_SetBgMode( 4 );            設定背景顯示模式為mode 4。
    ham_LoadBGPal(( void *)bg_Palette, 256 );        載入背景與物件調色盤。
    ham_LoadObjPal(( void *)object_Palette, 256 );
    ham_LoadBitmap (( void *)bg1_Bitmap);            載入背景1(遊戲初畫面)
    ham_FlipBGBuffer();                    將背景由緩衝區顯示至螢幕。
    for (kl=0;kl<900000;kl++){}            延遲遊戲初畫面。
    kl=0;                                  讓kl用來回收給gameover利用。
    ham_LoadBitmap (( void *)bg2_Bitmap);    載入遊戲說明頁(bg2)。
    ham_FlipBGBuffer();                    將背景由緩衝區顯示至螢幕。
    while(F_CTRLINPUT_B_PRESSED == 0){}    延遲遊戲說明頁。
    ham_PlaySample(mysample[0]);           發動背景音效。
    aiming = ham_CreateObj(( void *)red_aiming_anim_Bitmap,
             OBJ_SIZE_32X32 ,OBJ_MODE_NORMAL, 1  , 0 , 0 , 0 , 0 , 0 , 0
                ,aiming_x,aiming_y);
    ham_SetObjPrio(aiming,0);
        以上建立32x32大小的準心圖片，並將物件優先序號設為最高，不會被其他圖片覆蓋。
    for (i=0;i<base_hp;i++){
        heart[i] = ham_CreateObj(( void *)heart_Bitmap,OBJ_SIZE_16X16
                ,OBJ_MODE_NORMAL, 1  , 0 , 0 , 0 , 0 , 0 , 0 ,i*16,144);
    }
    for (i=0;i<6;i++){
        bbullet[i] = ham_CreateObj(( void *)bullet_Bitmap, OBJ_SIZE_16X8
            ,OBJ_MODE_NORMAL, 1  , 0 , 0 , 0 , 0 , 0 , 0 ,224,152-i*8);
```

```
    }
```

以上用兩個迴圈建立玩家生命的愛心圖片與玩家子彈數的圖片。

```
    eneapp();
```

發動敵人處理函式、可建立敵人。

```
    ham_LoadBitmap (( void *)water_Bitmap);
    ham_FlipBGBuffer();
```

載入遊戲中背景並顯示在螢幕上。

```
    ham_StartIntHandler(INT_TYPE_VBL,( void *)&vbl_func);
```

ham_StartIntHandler是ham內建的重要函數，傳入主要執行的函數，他會每六十分之一秒執行一次，其中我們在裡面使用k作為循環計量。

```
    while ( 1 ){
        if (F_CTRLINPUT_START_PRESSED) break;
    }
```

一個使程式不要結束的無窮迴圈，如果玩家按下start則會結束程式，然後程式會自動重新開始遊戲。

```
    return 0 ;
}
```

自定函示介紹：

```
void eneapp(){
    for (i=0;i<ENE_NUM;i++){
        ene[i] = ham_CreateObj(
    ( void *)ene_Bitmap,OBJ_SIZE_16X32,OBJ_MODE_NORMAL,1,2,0,0,0,1,0,5,5);
        ham_SetObjPrio(ene[i],0);
        ham_SetObjVisible(ene[i], 0);
    }
}
```

一、本函數是作為一個建立敵人的開頭。

二、由一個迴圈掃描每一個敵人(註：ENE_NUM是定義敵人人數的參數)，用陣列ene[]作為指向每一個敵人的指標，此後要對任何一號敵人做更動只要透過這個指標陣列即可。

三、Ham_CreateObj是ham已經定義的函式，將有另一頁專門介紹。

四、ham_SetObjVisible也是ham內部定義的函式，其中參數0代表看不見、1代表看的見；在此用其來建立敵人以後，將敵人預設為看不見。

五、ham_SetObjPrio用來將物件的優先順序設定到最高，以免被背景覆蓋。

```
void vbl_func() {
	ham_CopyObjToOAM();
	query_buttons();
	update_aiming_gfx();
	update_aiming_pos();
	if (blood_map>0)blood_map--;
	if (blood_map==0){
		blood_map=-1;
		ham_LoadBitmap (( void *)water_Bitmap);
		ham_FlipBGBuffer();
	}
	k++;
	if(k == 1001+ENE_ATK_TIME)    k = 1;
	if (k>(ENE_ATK_TIME+ENE_TIME-1))
		ene_atk();
	if (a_press_ava){
		base_bullet--;
		ham_SetObjVisible(bbullet[base_bullet],0);
	}
	set_obj_xy();
	if (!(k%ENE_TIME) && (k/ENE_TIME)<11)
		ene_appar((k/ENE_TIME)-1);
	ene_state();
	ham_CopyObjToOAM();
	ham_SyncMixer();
	ham_UpdateMixer();
	if(!mysample[0]->playing){
	    ham_PlaySample(mysample[0]);
	}
return ;
}
```

本函式是所有主要動作的驅動函式，在此引用的函式將會一一介紹，在此不多註。

一、ham_CopyObjToOAM()是ham的內建函式，用來將物件搬移到顯示記憶體。

二、query_buttons()是自訂的函式，用以檢測每個按鍵的情形。

三、update_aiming_gfx()是自訂函式，用來更新準心的圖片 。

四、update_aiming_pos()是自訂函式，用來更新準心的位置。

五、blood_map是一個整數計量，當玩家被敵人攻擊後他將顯示負值，這段期間背景將會改成被擊中的背景，過了這個計量就會被改為原本的背景。

六、由於主函式呼叫的ham_StartIntHandler每六十分之一秒會執行一次，這段期我們需要k作為循環計數，以上是寫k每經過1001+ENE_ATK_TIME次就會歸零重新計算，順但一提，本遊戲的敵人座標也試用k當作參數的。

七、每當敵人存在而且到攻擊時間時，使用ene_atk函數來實做敵人攻擊所產生的

情況。

八、下一段是在確認攻擊按鍵合法之後，用來減少螢幕上顯示的子彈數。(註：由於有特殊設定避免連射，所以以判定合法的射擊為主來更新其他動作)

九、set_obj_xy()是更新每一個敵人座標的自訂函式。

十、ene_appar()是敵人出現的函式，前段判別式用來實做先後出現敵人。

十一、 ene_state()用來更新敵人的狀態(在場上？被擊中？不在場上？)

十二、 ham_CopyObjToOAM()在次將更新過之後的各部份傳送到顯示記憶體。

十三、 最後四行是引用音訊函數的必要陳述。

```
void query_buttons() {
```

這裏query_buttons()函式相當的長，所以在此將每一個部分逐行解析，就不放在最後面以免對照麻煩。

```
    if (a_press_count    <AIMING_CONT)        a_press_count++;
    if (a_press_count==AIMING_CONT && F_CTRLINPUT_A_PRESSED &&
                                                                    base_bullet){
        a_press_ava=1;
        a_press_count=0;
    }else if (a_press_count==AIMING_CONT && !F_CTRLINPUT_A_PRESSED){
        a_press_count==AIMING_CONT;
    }else if (a_press_count < AIMING_CONT){
        a_press_ava=0;
    }
```

這裡使用a_press_count實做發射延遲，每當它在AIMING_CONT的值時按下A鍵後發射才有意義，不然時間間隔太近的輸入都會無效化。
這裡輸出的a_press_ava是有效發射，用來給往後的射擊做判定。

```
    if (a_press_ava )check_atk();
```

這裡將判定有效攻擊時，進入攻擊判定函式。

```
    if (F_CTRLINPUT_B_PRESSED && base_box>0 && base_bullet==0){
        base_box--;
        base_bullet=6;
        for (i=0;i<6;i++)      ham_SetObjVisible(bbullet[i],1);
    }
```

這裡判定按下B的時候是否補充子彈。

以下用來實做按下每個方向鍵時的準心變化，舉第一個為例：如果只按上其他不按，則y座標減一到零為止，dir_aiming參數以供更改準心圖片使用。
其他方向的原始碼則可一以貫之，在此不多贅述。

```
//UP only
    if (F_CTRLINPUT_UP_PRESSED && !F_CTRLINPUT_RIGHT_PRESSED
                                        && !F_CTRLINPUT_LEFT_PRESSED){
        if (aiming_y > 0 ) aiming_y--;
        dir_aiming = ANIM_UP;
        return ;
}
// UP + RIGHT
```

```
        if (F_CTRLINPUT_UP_PRESSED && F_CTRLINPUT_RIGHT_PRESSED){
            if (aiming_y > 0 ) aiming_y--;
            if (aiming_x < 208 ) aiming_x++;
            dir_aiming = ANIM_UPRIGHT;
            return ;
    }
    // RIGHT only
        if (F_CTRLINPUT_RIGHT_PRESSED && !F_CTRLINPUT_UP_PRESSED
                                        && !F_CTRLINPUT_DOWN_PRESSED){
            if (aiming_x < 208 ) aiming_x++;
            dir_aiming = ANIM_RIGHT;
            return ;
    }
    // DOWN + RIGHT
        if (F_CTRLINPUT_DOWN_PRESSED && F_CTRLINPUT_RIGHT_PRESSED){
            if (aiming_y < 128 ) aiming_y++;
            if (aiming_x < 208 ) aiming_x++;
            dir_aiming = ANIM_DOWNRIGHT;
            return ;
    }
    // DOWN only
        if (F_CTRLINPUT_DOWN_PRESSED && !F_CTRLINPUT_RIGHT_PRESSED
                                        && !F_CTRLINPUT_LEFT_PRESSED){
            if (aiming_y < 128 ) aiming_y++;
            dir_aiming = ANIM_DOWN;
            return ;
    }
    // DOWN + LEFT
        if (F_CTRLINPUT_DOWN_PRESSED && F_CTRLINPUT_LEFT_PRESSED){
            if (aiming_y < 128 ) aiming_y++;
            if (aiming_x > 0 ) aiming_x--;
            dir_aiming = ANIM_DOWNLEFT;
            return ;
    }
    // LEFT only
        if (F_CTRLINPUT_LEFT_PRESSED && !F_CTRLINPUT_UP_PRESSED
                                        && !F_CTRLINPUT_DOWN_PRESSED){
            if (aiming_x > 0 ) aiming_x--;
            dir_aiming = ANIM_LEFT;
            return ;
    }
    // UP + LEFT
        if (F_CTRLINPUT_UP_PRESSED && F_CTRLINPUT_LEFT_PRESSED){
```

```
            if (aiming_y > 0 ) aiming_y--;
            if (aiming_x > 0 ) aiming_x--;
            dir_aiming = ANIM_UPLEFT;
            return ;
        }
        return ;
}
```

到此按鈕的判定結束，回傳至vbl_func()已進行其他的功能函數使用。

```
void update_aiming_gfx() {
    if (a_press_ava)        visidual_count=15;
    else if (!a_press_ava && visidual_count)        visidual_count--;
    ham_UpdateObjGfx(aiming, ( void *)&red_aiming_anim_Bitmap[1024
                                                    *(dir_aiming+(visidual_count/8)*8)]);
    return ;
}
```

這個update_aiming_gfx()用來將query_buttons()修改的aimimg作更新準心的依據。

一、如果是有效攻擊，則延遲計數設為15，準心圖像轉為顯示發射中的圖示，引用延遲計數是因為如果每1/60秒更新回原本的準心，人類眼睛視覺暫留不夠久會看不到。

二、如果非有效攻擊中，則堅延遲計數遞減。

三、當延遲計數低於8時，傳入red_aiming_anim_Bitmap的參數則為原本待命中的準心位置，以15和/8就可以以一個參數實做視覺暫留，而不需要兩個。

```
void update_aiming_pos() {
    ham_SetObjX(aiming,aiming_x);
    ham_SetObjY(aiming,aiming_y);
    return ;
}
```

這個update_aiming_pos()用來更新由剛剛query_buttons()更新的準心的位置，將新的XY值輸入這個函數，將引用到ham內建的函式ham_SetObjX和ham_SetObjY，可以將物件的座標值更新。

```
void ene_atk(){
    for (i=0;i<ENE_NUM;i++){
        if (k >= (ENE_ATK_TIME+ENE_TIME*(i+1)) && ene_exist[i]!=0 &&
                                        ((k-ENE_ATK_TIME) % ENE_TIME)==0 ){
            base_hp--;
            if (base_hp<=0)    {
                gameover();
                ham_StopIntHandler(( void *)&vbl_func);
            }
            ham_DeleteObj(heart[base_hp]);
            ene_count_x[i]=-20;
            if (base_hp>0)
                blood_map=10;
        }
    }
    if (blood_map >0){
        ham_LoadBitmap (( void *)water_atked_Bitmap);
        ham_FlipBGBuffer();
    }
}
```

這個函數實做敵人的攻擊。

一、當敵人出現後只要他還存在(ene_exist[i]>0)每經過ENE_TIME的時間玩家的生命值就會減少，減少螢幕顯示的愛心數、敵人改變成開火狀態圖示。

二、將ene_count_x[i]設為-20一樣由同一個變數實做視覺暫留，由於其他判別只用到0以及1，將負號利用做為視覺暫留的計數。

三、Blood_map實做遭受攻擊後背景會變化，也是一個計數，當玩家遭受攻擊他會改成正值遞減，地檢期間是為視覺暫留，不跟敵人一起計數是因為一個時間點多的及人攻擊時，只要載入一次地圖即可。

四、當玩家生命值低於零時，呼叫gameover函數，進入遊戲結束階段。

```
void set_obj_xy(){
   ene_x[0]=abs((k%750)-375)*224/375;
ene_y[0]=abs((k%600)-300)*128/300;
   ene_x[1]=224-abs((k%600)-300)*224/300;
      ene_y[1]=abs((k%750)-375)*128/375;
   ene_x[2]=30+abs((k)%360-180)*140/180;
ene_y[2]=0+abs((k+90)%360-180)*100/180;
   ene_x[3]=50+abs((k+180)%360-180)*140/180;
                                        ene_y[3]=20+abs((k+270)%360-180)*100/180;
   ene_x[4]=224-(k%900)*224/900;                          ene_y[4]=100;
   ene_x[5]=230-(abs((k%750)-375))*200/375;
                                     ene_y[5]=abs(160-(abs((k%450)-225))*200/225);
   ene_x[6]=abs((k%450)-225)*224/225;
ene_y[6]=abs((k%750)-375)*128/375;
   ene_x[7]=224-abs((k%600)-300)*224/300;
      ene_y[7]=abs((k%450)-225)*128/225;
   ene_x[8]=abs((k%450)-225)*224/225;
      ene_y[8]=abs((k%600)-300)*128/300;
   ene_x[9]=10;                                           ene_y[9]=10;
}
```

這個函數存放k值與敵人座標的相依關係。

```
void check_atk(){
      ham_PlaySample(mysample[1]);
      for (i=0;i<ENE_NUM;i++){
            if (ene_x[i]-18 < aiming_x && aiming_x < ene_x[i]+2 && ene_y[i]-18 <
                                                aiming_y && aiming_y < ene_y[i]+18){
                  ham_UpdateObjGfx(ene[i], ( void *)&ene_Bitmap[2560]);
                  base_score++;
                  ene_exist[i]=-7;
            }
      }
}
```

這個函數用來判定有效攻擊後敵人的狀況。

一、每次有效攻擊的時候，如果準心打中對的位置，此位置相對應的敵人物件將會改成被擊中的圖示。

二、一樣以ene_exist[i]的負數部份實做視覺暫留一段時間以後將敵人設為看不見，然後擊倒敵人數增加一。

三、ham_PlaySample是引用聲音函數，在這裡當有效發射成功時，引入mysample[1]的聲音文件。

```
void ene_state(){
    for (i=0;i<ENE_NUM;i++){
        if (ene_exist[i]>0){
            ene_count_x[i]++;
            if (ene_count_x[i]==50)ene_count_x[i]=0;
            if (ene_count_x[i]>=0){
                if (ene_pre_x[i]<ene_x[i])
                    ham_UpdateObjGfx(ene[i], ( void *)
                                        &ene_Bitmap[(ene_count_x[i]/25)*512]);
                else if(ene_pre_x[i]>ene_x[i])
                    ham_UpdateObjGfx(ene[i], ( void *)
                                    &ene_Bitmap[1024+(ene_count_x[i]/25)*512]);
            if (ene_count_x[i]<0){
                if (ene_x[i]<119)ham_UpdateObjGfx(ene[i], (void *)
                                                        &ene_Bitmap[2048]);
                else ham_UpdateObjGfx(ene[i], (void *)&ene_Bitmap[3072]);
            }
            ham_SetObjXY(ene[i],ene_x[i],ene_y[i]);
            ene_pre_x[i]=ene_x[i];
        }
        if (ene_exist[i]<0)     ene_exist[i]++;
        if (ene_exist[i]==0){
            ene_disapp(i);
        }
    }
}
```

這個函數用來判斷每個位置物件的讀取情形，用來實做敵人轉向與開火。

一、ene_exist 視每個敵人的存在情形。當他是負的則用做視覺暫留計量，0表示他被擊倒或未出現。

二、ene_count_x是專屬於每一個敵人的計量，正負值剛好當不同的計量數。

三、ene_pre_x　ene_x是敵人前一個x座標與現在的x座標，當前一個座標小於現在，表示敵人向右走，反之則向左走，引入ene_count_x當作在更新圖檔的時候的參數，除以25正好實作0到49之間的01布林值。

四、在敵人開火的時候ene_count_x變為負值，再此當作另一個計量；當敵人的x座標大於119則敵人朝左向玩家開火，小於119則朝右向玩家開火。

```
void ene_appar(u8 j){
	ham_SetObjVisible(ene[j], 1);
	ene_exist[j]=1;
}

void ene_disapp(u8 j){
	ham_UpdateObjGfx(ene[i], ( void *)&ene_Bitmap[0]);
	ham_SetObjVisible(ene[j], 0);
}
```

這兩個函數更改敵人物件的狀態，引用兩個內建函數並入ene_exist的修改。

一、出現敵人時同時讓敵人的狀態改成1(存在)，然後將敵人設為可看見。

二、敵人要消失時便將敵人設為看不見，並把敵人的圖示改成檔案的出始狀態。

三、這裡使用出現與隱藏是因為敵人會隨著k的循環而再次出現，才不會使得讀寫記憶體一直進行。

```
void gameover(){
    if (kl==0){
        ham_LoadBitmap (( void *)gameover_Bitmap);
        ham_FlipBGBuffer();
        ham_DeleteObj(aiming);
        base_score*=base_score;
        base_score*=48;
        for (i=0;i<11;i++){
            number_pic[i]=ham_CreateObj(( void *)num_Bitmap,
            OBJ_SIZE_8X8 ,OBJ_MODE_NORMAL, 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 );
            if (i>0){
                ham_UpdateObjGfx( number_pic[i] ,
                                        ( void *)&num_Bitmap[0+(64*(i-1))] );
                ham_SetObjVisible( number_pic[i] , 0) ;
            }
        }
        for (i=0;i<10;i++){
            if (i<6)ham_DeleteObj(bbullet[i]);
            ham_DeleteObj(ene[i]);
        }
        i=0;
        while(base_score>0){
            ham_SetObjXY ( number_pic[base_score%10+1] , 140-i*8 , 120 );
            ham_SetObjVisible (number_pic[base_score%10+1] , 1 );
            i++;
            base_score/=10;
        }
        kl++;
    }
}
```

這個函數是當玩家生命值歸零時所呼叫的，用來決定遊戲的結束狀態。

一、kl是先前顯示遊戲開始畫面的時候使用的計數，在這裡回收作為gameover的計數，因為ham會根據時間一直呼叫函數，使用kl=1設定只讓gameover執行一次。

二、進入gameover則將背景轉換成gameover的背景。

三、然後使用 ham_DeleteObj函數來依次把所有的物件移除（彈藥、敵人）。

四、由於ham不推薦我們使用文字模式，所以在這裡我們使用圖片物件來實作算分系統。Num圖片是0到9的文字圖形，我們依次把每個數字放到number_pic陣列裡面，注意number_pic[1]從圖片0開始，前一格留給系統做記憶體緩衝。

這次聲音的部份是用HAM內建的Direct Sound Control Function（DSCF）來實做。之前有考慮過用document推薦的Krawall來作，不過網路上的參考資料似乎不多，HAM本身給的說明又不是很清楚，所以還是選擇了DSCF

不過要用DSCF，就必須用wav2gba將wav檔轉成gba可以接受的格式，但wav2gba只接受某特定格式的wav檔（ACM waveform，filter PCM），因此必須先將要用的音效用cool edit pro轉檔才行。

此外還必須修改makefile

在OFILES += final.o 這行的後面加入bgm.o sound1.o sound2.o sound3.o（所有的音效檔名.o）

這樣HAM在compile時會自動用wav2gba把我們要用的音效檔轉檔。

```
extern const WaveData _binary_bgm_raw_start;
extern const WaveData _binary_sound1_raw_start;
extern const WaveData _binary_sound2_raw_start;
extern const WaveData _binary_sound3_raw_start;
```

建立與音效檔關聯的struct

以下是mygba.h中wavedata 這struct的內容

```
typedef struct {
    u16 type;     //Type
    u16 stat;     //Status
    u32 freq;     //Frequency Calculation Value
    u32 loop;     //Loop Start Point
    u32 size;     //Sampling Number
    s8 data[1]; //Sampling Data Order
} WaveData
    mysample[0] =   ham_InitSample((u8*)_binary_bgm_raw_start.data,
                     _binary_bgm_raw_start.size,
                     _binary_bgm_raw_start.freq>>10);              //bgm
    mysample[1] =   ham_InitSample((u8*)_binary_sound1_raw_start.data,
                     _binary_sound1_raw_start.size,
                     _binary_sound1_raw_start.freq>>10);         //射擊音效
    mysample[2] =   ham_InitSample((u8*)_binary_sound2_raw_start.data,
                     _binary_sound2_raw_start.size,
                     _binary_sound2_raw_start.freq>>10);         //被打的音效
    mysample[3] =   ham_InitSample((u8*)_binary_sound3_raw_start.data,
                     _binary_sound3_raw_start.size,
                     _binary_sound3_raw_start.freq>>10);         //gamever
```

建立可以播放的音效

之後就可以用ham_PlaySample來播放音效了

```
ham_SyncMixer();
ham_UpdateMixer();
```

這兩個funciton每個frame都要執行一次

```
if(!mysample[0]->playing){
    ham_PlaySample(mysample[0]);
}
```

每frame檢查一次，如果背景音樂播完了就再放一次

計算機組織與組合語言期末報告
第一次會議紀錄

組員：王舜玄　林仲丘
會議日期：十二月二十四日下午六時三十分
會議地點：資訊系館地下室
會議重點：

一、概略構想期末報告為在HAM上實做一個GBA遊戲。
二、尋找並確認組員的興趣與想法。
三、分配組員應尋找的資料。
四、給組員提要並且互相確認應在做報告以前應有的基本智識。
五、請全體組員期末考加油，以期早日開始完成期末報告。

計算機組織與組合語言期末報告
第二次會議紀錄

組員：王舜玄　林仲丘
會議日期：一月二十日早上九時三十分
會議地點：資訊系館地下室
會議重點：

六、確認期末報告為在HAM上實做一個GBA射擊遊戲。
七、其遊戲內容為遊戲者以上下左右鍵控制準心，螢幕上將隨機出現敵人並且隨機移動，操縱者必須移動準心與控制板機攻擊敵人。
八、如果敵人存活超過一段時間便會開始攻擊遊戲者，遊戲者有基本生命值，遭受太多攻擊會結束遊戲。
九、或許會引入限制子彈與換子彈夾的功能。
十、打擊敵人數、使用子彈數、生命消耗量會計入總分計算。
十一、　開始學習HAM的基本結構、如何在HAM上面的基本操作：例如讀取圖片檔案、以鍵盤按鍵移動圖案、時間計數與動態圖片讀取。
十二、　本日使用的參考資料：

中國遊戲開發論壇
http://bbs.ogdev.net/TopicContent.aspx?BoardID=2&TopicID=6177

HAM教程翻譯
http://angeljicu.googlepages.com/ham

計算機組織與組合語言期末報告
第三次會議紀錄

組員：王舜玄　林仲丘
會議日期：一月二十一日早上九時三十分
會議地點：資訊系館地下室、台大男七舍
會議重點：

十三、　將昨日分配的進度做完並互相驗收。
十四、　實做鍵盤移動圖片與圖片自動出現並移動。
十五、　實做準心根據不同方向會變色。
十六、　實做準心發射後延遲避免連續發射。
十七、　遊戲圖片做圖。
十八、　討論遊戲基本設定值：生命值、彈藥數等等。
十九、　ＧＦＸ２ＧＢＡ指令的進階了解。
二十、　創造物件、刪除物件、設定物件優先序、隱藏以及顯示物件指令應用。
二十一、　開始尋找音效的實做方式。
二十二、　本日使用的參考資料：

中國遊戲開發論壇
http://bbs.ogdev.net/TopicContent.aspx?BoardID=2&TopicID=6177

HAM教程翻譯
http://angeljicu.googlepages.com/ham

ＨＡＭ　Ｄｏｃｕｍｅｎｔａｔｉｏｎ
http://www.ngine.de/hamdoc/index.html

計算機組織與組合語言期末報告
第四次會議紀錄

組員：王舜玄　林仲丘
會議日期：一月二十二日早上九時三十分
會議地點：資訊系館地下室、台大男七舍
會議重點：

二十三、　將昨日分配的進度做完並互相驗收。
二十四、　改善圖片視覺延遲的實做方式。
二十五、　實做自行移動圖片與動態圖片顯示。
二十六、　建立目標自行移動之座標函數。
二十七、　目標會隨著左右移動方向不同而轉左轉右。
二十八、　目標會隨著移動而有腳步動作。
二十九、　目標攻擊玩家時會顯示開火。
三十、　目標攻擊玩家時會因為在螢幕左右不同而轉頭開火。
三十一、　實做玩家的生命值與彈藥值。
三十二、　製作開頭與結束畫面。
三十三、　尋找音效的實做方式未果。
三十四、　本日使用的參考資料：

中國遊戲開發論壇
http://bbs.ogdev.net/TopicContent.aspx?BoardID=2&TopicID=6177

HAM教程翻譯
http://angeljicu.googlepages.com/ham

ＨＡＭ　Ｄｏｃｕｍｅｎｔａｔｉｏｎ
http://www.ngine.de/hamdoc/index.html

Ｅｍｕｌａｔｉｏｎ　ＨＱ
http://www.emuhq.com/idx/63/0/VisualBoyAdvance-Section.html

計算機組織與組合語言期末報告
第五次會議紀錄

組員：王舜玄　林仲丘
會議日期：一月二十三日早上九時三十分
會議地點：資訊系館地下室、台大男七舍
會議重點：

三十五、　將昨日分配的進度做完並互相驗收。
三十六、　改善人物座標計算順序。
三十七、　實做玩家受攻擊後變換背景。
三十八、　實做聲音函式。
三十九、　建立聲音檔案轉換格式。
四十、　在動作內引用聲音函式。
四十一、　新增開頭說明與版權頁。
四十二、　新增遊戲結束欄。
四十三、　實做reset。
四十四、　使用圖片檔案製作計數。
四十五、　建立計算得分數機制。
四十六、　得分數以圖片顯示。
四十七、　本日使用的參考資料：

中國遊戲開發論壇
http://bbs.ogdev.net/TopicContent.aspx?BoardID=2&TopicID=6177

HAM教程翻譯
http://angeljicu.googlepages.com/ham

ＨＡＭ　Ｄｏｃｕｍｅｎｔａｔｉｏｎ
http://www.ngine.de/hamdoc/index.html

Ｅｍｕｌａｔｉｏｎ　ＨＱ
http://www.emuhq.com/idx/63/0/VisualBoyAdvance-Section.html

結語：

組合語言是本學期唯一的系定必修程式課程，在本課程的學習中我終於了解並且對一個以前的疑惑有了初步的認識：我們寫的程式碼與硬體顯示的部份到底是怎麼樣子溝通的？這些在學期初的電路分析與玩具機械有了初步的認識，然後再學到怎麼直接去操縱記憶體與讀寫輸入的組合語言，這一連貫的課程讓我們這些大學才開始專研電腦的"弱者"們有了更精進的認識。

本次的期末報告在教授公佈形式以後，本人便義無反顧的堅持要做遊戲。想當年，電動遊戲是許多人的共同回憶，從一開始的掌上型俄羅斯方塊，到人生的第一台任天堂紅白機，那些整天盯著螢幕弄到差一點近視的歲月裡陪伴著我們的都是遊戲，那些程式設計師的心血結晶。直到小學，也在補習班為了和同學搶位置玩模擬城市與仙劍奇俠傳而大打出手，遊戲可以使人沈浸在那種動作、聲光與劇情的氣氛中，可以說就是一部部有聲音、有影像的電子小說，還記得在補習班一群人在嘗試著各種做法終於打敗拜月教主的時候，看到那種結局不知道多少男生女生當場飆淚。於是在許久的當年，便在心裡默默的許下一個心願：希望以後我的作品也可以這樣子感動人。

也不知道過了多久，也不知道多久都沒有再回想起的宏願。過了國中與高中，慶幸自己有這種機會選擇也榮幸的接受資訊系的選擇而來資訊系報到，去年計概的課要我們破解遊戲的存檔，才黯然的想起為什麼當初淺意識中先選擇資訊系的理由，看著18歲的逍遙哥哥已經變成弟弟了，更覺得自己應該趕快紮實的學成，才能趕快的完成人生的夢想。

期末考結束以後，終於有了完整的時間可以將規劃了一個月的想法實現，礙於人數的限制沒有辦法做出精彩絕倫的遊戲，但是我們願意在每一個小細節投入心血，使得作品不要有任何瑕疵。

我們第一個遇到的問題就是<mygba.h>裡面定義的函式運作完全不熟悉，除了去看原始碼以外我們也上網找了許多參考資料，幸好同組同學簡體中文辨識能力極佳，因為我看簡體會比英文慢（笑）。

再來也是很關鍵的就是兩個人的成長背景不同，連帶著想法與寫出來的程式碼會完全不相容，幸好我們有上學期資訊系統原理學到的許多整合的概念，可以讓不同實作方式的程式碼結合在一起，而對於<mygba.h>許多函式的用法除了找使用說明以外，也會看看網路上其他網友的使用方法，以期在現有的方法中更加延伸另一種不同的境界。

最後一個問題也是延伸到報告時候的問題，聲音檔案的格式我們不僅一開始找不到，連怎麼轉換都不知道，另一位組員也是投入了相當多的時間去諮詢別人與尋找資料，最後做出在電腦上還算穩定的音效檔案，但是放到GBA不能運作。

在本次的期末報告中，我學到的也是資訊系一貫的團隊合作精神，也是一種對自己的作品那種認真負責的態度，那種與BUG奮戰到你死我活的時光或許會是往後人生值得回味的一部份，今天報告結束了，謝謝老師與助教這一學期的指導與幫忙。

2008/01/24 王舜玄