

# 計算機程式 期末專題報告

B10901018 - 吳昶勳

組別：第五組(Escape The Loop)

RACE TO START

# 第一部分

# 專題企畫

剛開始我們還在為遊戲發想的階段時，我們參考了許多現今已上架到網路平台上的遊戲，考量到許多可行性與複雜度的問題之後，我們決定最終以「元氣騎士」作為藍圖，設計一款2D操作型遊戲。

遊戲一開始玩家扮演一隻小地精出發，遊戲目標就是為了逃脫這個看似沒有盡頭的迴圈，若是成功討伐最終的大魔王，即可獲勝。

一路上會遇到一些敵人與商店，商店能夠給予武器效能、基本生命值等各種幫助；相對的，越接近路途尾聲敵人也越難纏，祝福每隻遠征的小地精都能夠完成艱困的任務平安歸來。

# 功能設計

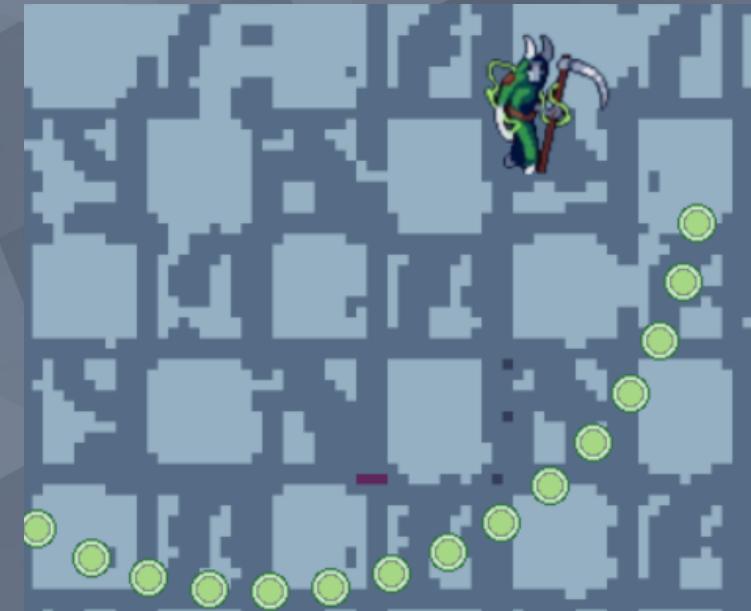
1. 主角：能透過WASD操控移動，點擊滑鼠瞄準並攻擊，上述行為皆有程式控制播映對應的動畫。  
(Character.cpp、Character.h)



# 功能設計

2.敵人：具備事先輸入之行為模式，能夠以特定方式移動並對玩家進行攻擊，同樣有程式控制播映不同怪物對應的動畫。

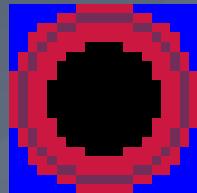
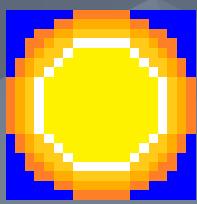
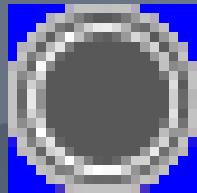
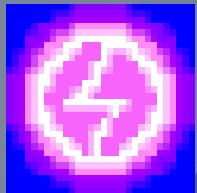
(Monster.cpp、Monster.h)



# 功能設計

3.子彈：接受玩家和怪物的指示，放出特定的子彈。有參數控制最多存在幾顆、飛行速度、攻擊力、碰到牆壁是否反彈、是否追蹤玩家。

(`Bullet.cpp`、`Bullet.h`、`Monster_Bullet.cpp`、`Monster_Bullet.h`)



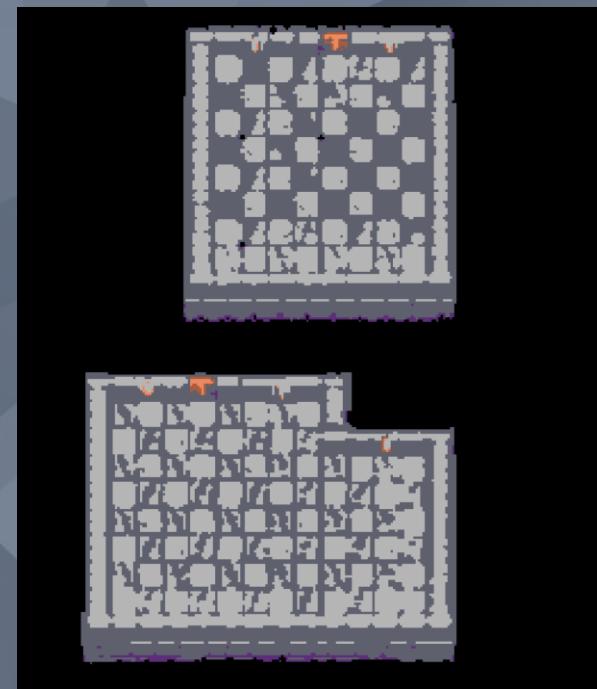
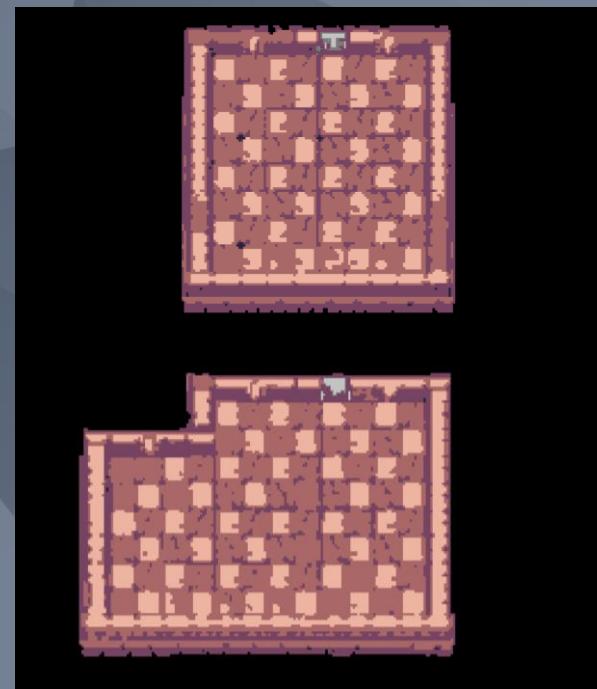
4.判定：判定主角子彈是否超出射擊上限、是否碰到怪物，怪物子彈是否碰到主角，判定腳色是否死亡並進行死亡處理。

(在主角與敵人Class內都有)

# 功能設計

5. 地圖：在不同層(3層)內有不同樣式，戰鬥與非戰鬥關卡亦有不同設計，邊界也防止角色超出地圖外。

(Map.cpp、Map.h)



# 功能設計

6.商店：滑鼠的游標移動到對應商品上有對應效果回饋，改變武器參數或玩家血量。

(Store.cpp)



# 功能設計

7. 轉場：判斷場上怪物是否清零，達成後判斷玩家是否走到門口，若皆達成則跳出圖示並進入下一關。 (Next.cpp)
8. 操作介面：遊戲開始圖示、當遊戲結束的判定下達，則給出對應圖片回饋。 (User\_Interface.cpp)



# 功能設計

9. 音樂音效：於遊戲開始時播放主題音樂、BOSS關播放魔王音樂、玩家受擊以及攻擊時播放音效。(於GlobalSetting.cpp和Character.cpp內)

10. 文字顯示：遊戲關卡與玩家血量之顯示，玩家受擊時閃爍回饋。

(Text.cpp、Text.h)



# 功能設計

11. 帧率控制：透過SDL計時器，控制畫面刷新速度。

(Ltimer.cpp、Ltimer.h、main.cpp內的timer\_set function)

12. 共用母結構：物件基本需要的內容。

(Basic.cpp、Basic.h)

```
class LTimer
{
public:
    //Initializes variables
    LTimer();
    //The various clock actions
    void start();
    void stop();
    void pause();
    void unpause();
    //Gets the timer's time
    Uint32 getTicks();
    //Checks the status of the timer
    bool isStarted();
    bool isPaused();
private:
    //The clock time when the timer started
    Uint32 mStartTicks;
    //The ticks stored when the timer was paused
    Uint32 mPausedTicks;
    //The timer status
    bool mPaused;
    bool mStarted;
};
```

```
void timer_set()
{
    //set delay
    int t = Frame_timer.getTicks();
    if(t < 20) SDL_Delay(20 - t);

    if(++timer==800) timer=0;
}
```

```
class basic{
public:
    virtual bool loadFromFile( std::string path,
    virtual void render(SDL_Rect * = NULL);
    void free();
    int getWidth();
    int getHeight();
    int getPos_x();
    int getPos_y();

protected:
    SDL_Texture * Texture;
    int Width;
    int Height;
    float Pos_x;
    float Pos_y;
    float Vel_x;
    float Vel_y;
};
```

# 美術設計

1. 開始畫面、轉場、結束(含成功與失敗)：背景是從網路上下載，再用 Canva 後製字幕，並利用彎曲、陰影、飄移等文字效果增加豐富度。



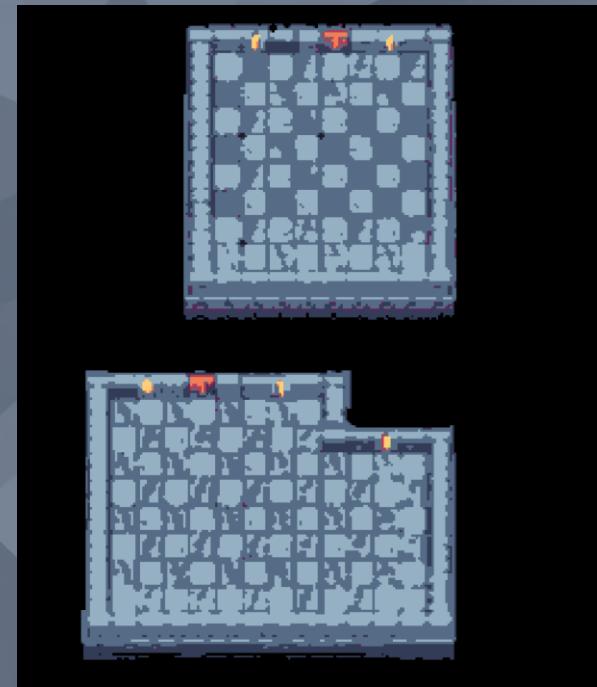
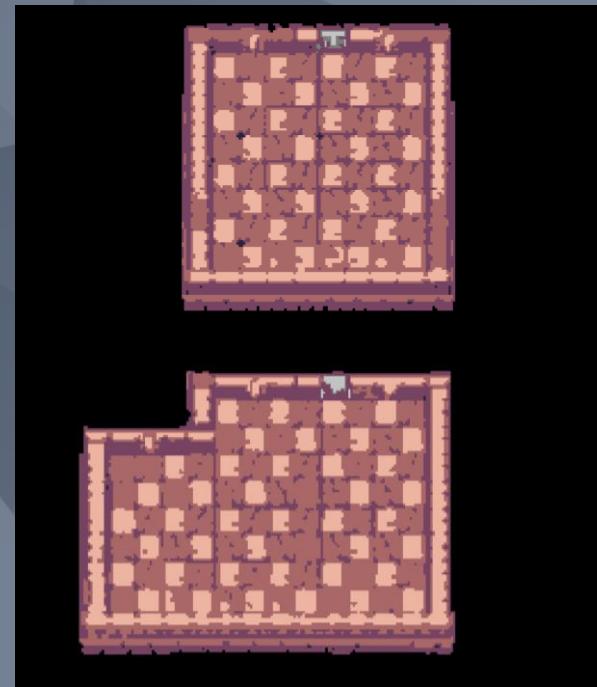
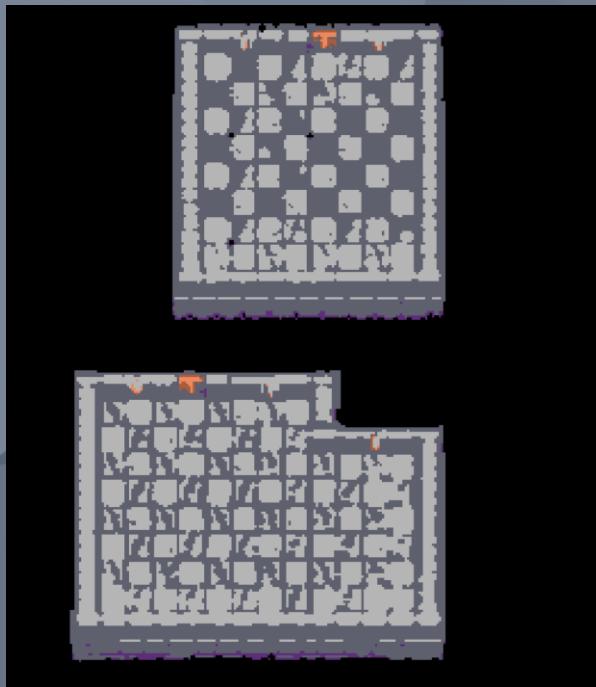
# 美術設計

2.商店：商店介面的圖層可分為背景以及選框。同樣是由網路上尋找現成素材，再經由小畫家及Canva編輯。為了能讓使用者能確認滑鼠游標是否已選到正確的道具上，製作了兩種顏色的邊框，讓滑鼠游標移至長方形內時得變換顏色，也因此選擇讓選框與背景分開為不同的圖層。



# 美術設計

3. 地圖：地圖是由網路上尋找材質，再利用小畫家裁減、拼貼，形成不同形狀的地圖，再透過像素轉換象素化，以及色階的轉換製作出有三種不同顏色，分別代表三層主要關卡。



# 美術設計

4. 角色連續動畫：角色的動作是由分格動畫組合而成。同樣從網路找尋現成素材，將每個分格動作切割成相同像素大小，再將同組動作拼成同一橫排，最後將同一位角色的所有動作合成為一張圖。



# 工作分配

## 1. 程式部分：

鄒昀樺：`User_Interface.cpp`、`Monster.cpp(0-319)`、`Monster.h`、遊戲音效(於`GlobalSetting.cpp`和`Character.cpp`內)。

李國瑋：`Monster.cpp(319-473)`

吳昶勳：`Bullet.cpp`、`Bullet.h`、`Character.cpp`、`Character.h`、`GlobalSetting.cpp`、`Ltimer.cpp`、`Ltimer.h`、`Map.cpp`、`Map.h`、`Monster_Bullet.cpp`、`Monster_Bullet.h`、`Next.cpp`、`Store.cpp`、`Text.cpp`、`Text.h`、`Basic.cpp`、`Basic.h`、`main.cpp`

# 工作分配

## 2. 美術部分：

鄒昀樺：主角、敵人、始末畫面、轉場、商店。

吳昶勳：地圖、子彈。

# 第一部分

A.

## Class架構圖

Level-1

Basic

Text

Ltimer

Level-2

Bullet

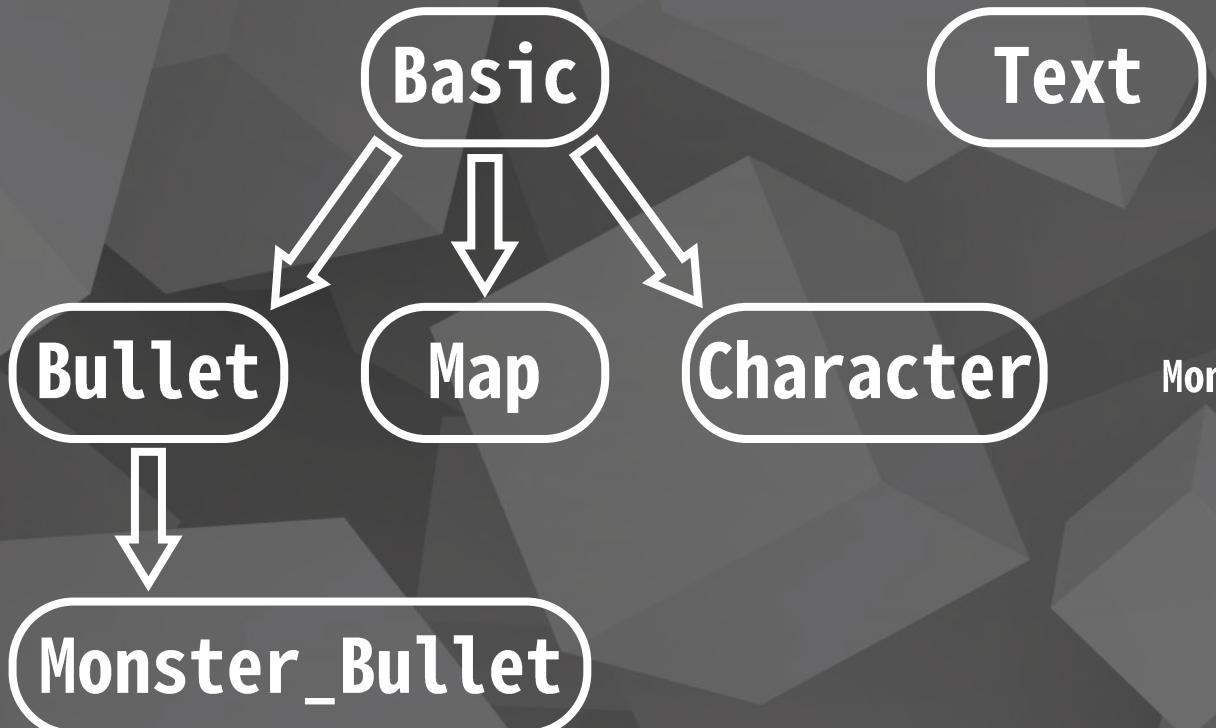
Map

Character

Monster(非本人撰寫)亦繼承Basic

Level-3

Monster\_Bullet



# A.

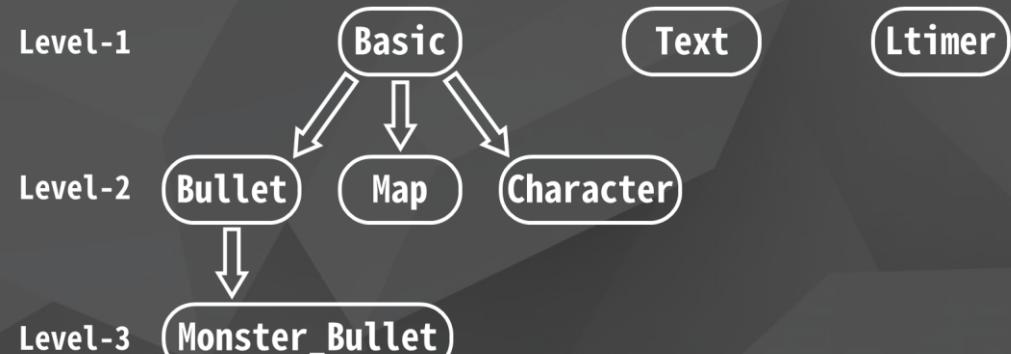
功能：

1. Basic : (同名.cpp.h)

基本要素如物件長寬、位置、速度、  
Texture等泛用性高之變數。

函式則包含loadFromFile(讀取圖片)、  
render(渲染物件)、free(去除舊材質)、  
get(長寬、位置)等泛用性高函式。

Class架構圖



```
class basic{
public:
    virtual bool loadFromFile( std::string path, Uint8 = 0x00, Uint8 =
    virtual void render(SDL_Rect * = NULL);
    void free();
    int getWidth();
    int getHeight();
    int getPos_x();
    int getPos_y();

protected:
    SDL_Texture * Texture;
    int Width;
    int Height;
    float Pos_x;
    float Pos_y;
    float Vel_x;
    float Vel_y;
};
```

# A.

功能：

2. Bullet : (同名.cpp.h)

繼承Basic之功能，負責主角子彈控制。

set\_v(接收主角指令設定子彈使飛行)、

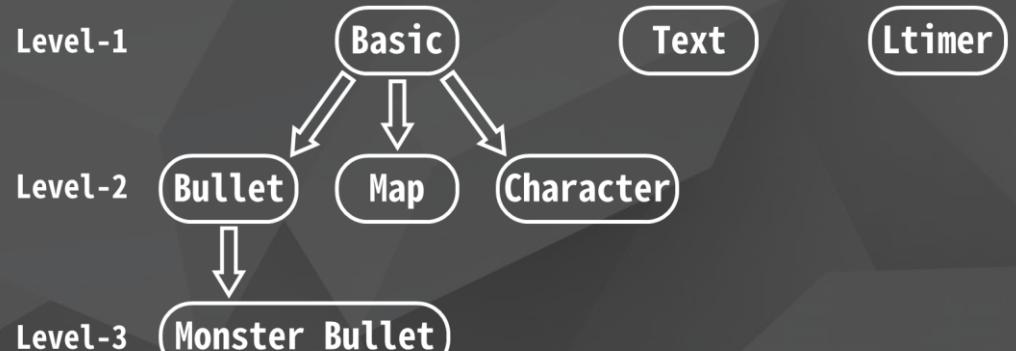
move(移動需要移動的子彈)、

check\_edge(檢查子彈是否觸碰邊界)、

check\_hit(檢查子彈是否擊中敵人)、

buff(接收指令後增強子彈)。

Class架構圖



```
class Bullet : public basic
{
    //friend Character;
public:
    //Initializes variables
    Bullet();
    //Deallocates memory
    ~Bullet();
    //Loads image at specified path
    virtual void set_v(int, int );
    virtual bool move();
    virtual void check_edge();
    virtual bool check_hit(int , int);
    void buff(int, int);
};
```

# A.

功能：

3. **Monster\_Bullet**：(同名.cpp.h)

繼承**Bullet**之功能，負責怪物子彈控制。

**set\_v**、**move**、**check\_edge**、**check\_hit**

之功能大致不變，覆寫惟細節處做修改。

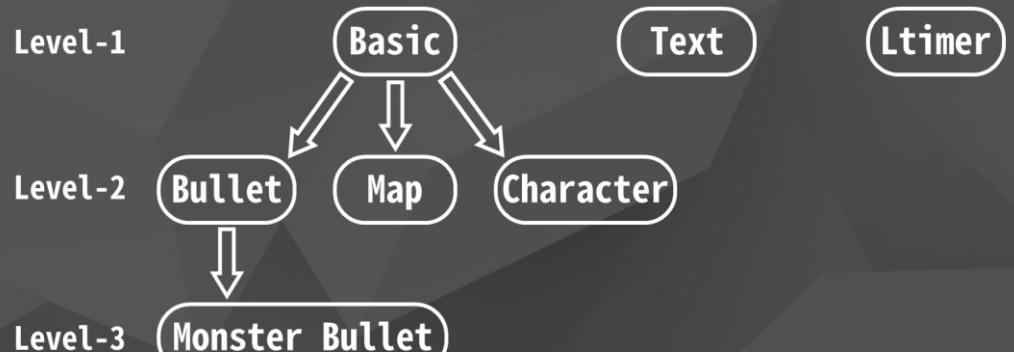
**set\_bullet**(更改子彈數值與材質)、

**change\_v**(透過**track**與否設定追蹤功能)、

**bullet\_bounce**(控制子彈是否反彈)、

**bullet\_type**(儲存子彈型態)等。

Class架構圖



```
class Monster_Bullet : public Bullet
{
public:
    Monster_Bullet();
    virtual void set_v(int, int, int = 0, float = 1);
    virtual bool check_hit(int x, int y);
    virtual void check_edge();
    virtual bool move();
    void set_bullet(bool, int, int, int);
    void change_v();

    bool track;
    bool bullet_bounce;
    int get_damage();

private:
    int bullet_type;
    int speed;
    int damage;
};
```

# A.

功能：

4. Map：(同名.cpp.h)

繼承Basic之功能，負責地圖控制。

因地圖檔案是一層三種在同一張，故覆寫

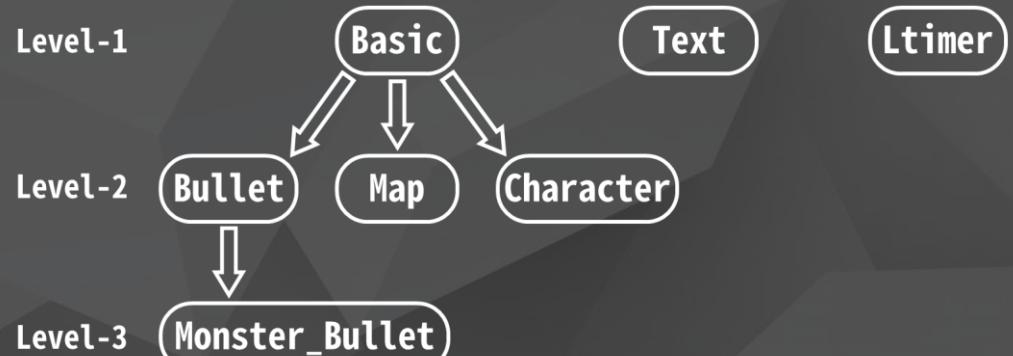
loadfromfile(因多圖而重新設定高度)、

operator~(進行渲染的工作)。

Collision\_check(提供需要使用之物件

進行簡單之邊界檢查)

Class架構圖



```
class Map : public basic
{
public:
    //Initializes variables
    Map();
    //Deallocates memory
    ~Map();
    //Loads image at specified path
    virtual bool loadFromFile( std::string path );
    virtual void render();
    //detect collision
    bool collision_check( float & x, float & y );

    void operator~ ();
};
```

# A.

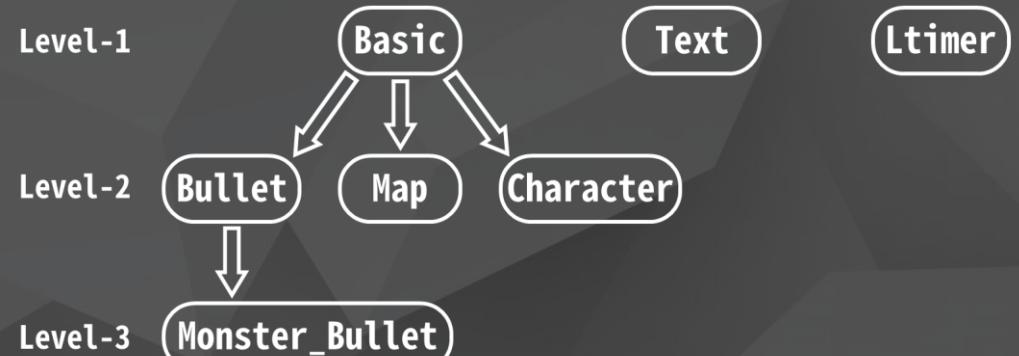
功能：

5.Character：(同名.cpp.h)

繼承Basic之功能，負責角色控制。

因角色動畫多圖合併在同一張，故覆寫  
`loadFromFile`(因多圖而重新設定長寬)、  
`render`(因動畫顯示須特殊設定Clip)。  
`move`(鍵盤移動)、`attack`(滑鼠攻擊)、  
`set_pos`(進入關卡設定角色位置)、  
`change_hp`(改變血量(Character\_Hp))、  
`Character_SpriteClips`(儲存動畫Rect)。

Class架構圖



```
class Character : public basic
{
public:
    //constructor
    Character();
    //destructor
    ~Character();
    virtual bool loadFromFile(std::string path, int = 8, int = 2);
    //Renders texture at Character position
    virtual void render();
    //change character position and sprite by keyboard event
    void move(SDL_Event*);
    //attack
    bool attack(SDL_Event*);
    //initialize characetr position
    void set_pos();
    //Gets image dimensions
    bool change_hp(int);
    int get_hp();
private:
    SDL_Rect * Character_SpriteClips;
    int      Character_Hp;
};
```

# A.

功能：

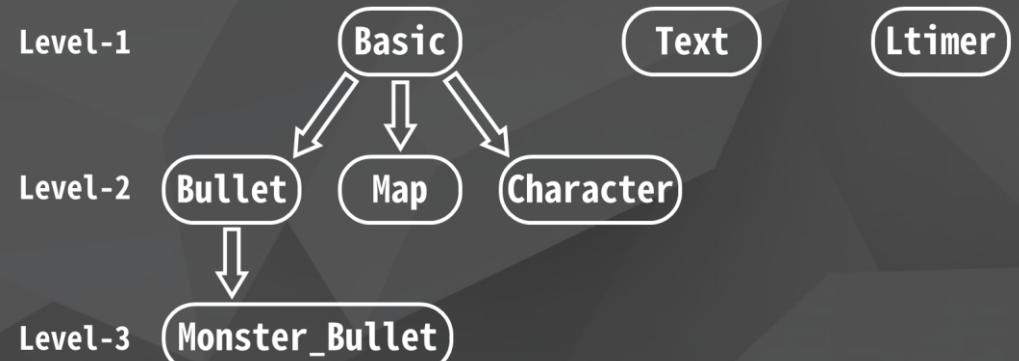
6.Text : (同名.cpp.h)

負責文字顯示，如角色血量、  
進行之關卡以及商店商品說明。

7.Ltimer : (同名.cpp.h)

負責記錄程式執行速度  
以便幀率控制。

Class架構圖



```
class Text
{
public:
    //Initializes variables
    Text();
    //Deallocates memory
    ~Text();
    //Creates image from font string
    bool loadFromRenderedText( std::string textureText, SDL_Color textColor );
    bool loadFromRenderedText2( std::string textureText, SDL_Color textColor );
    //Deallocates texture
    void free();
    //Set color modulation
    SDL_Color setColor( Uint8 red, Uint8 green, Uint8 blue );
    //Renders texture at given point
    void render( int x, int y );
private:
    //The actual hardware texture
    SDL_Texture* mTexture;
    int mWidth;
    int mHeight;
};
```

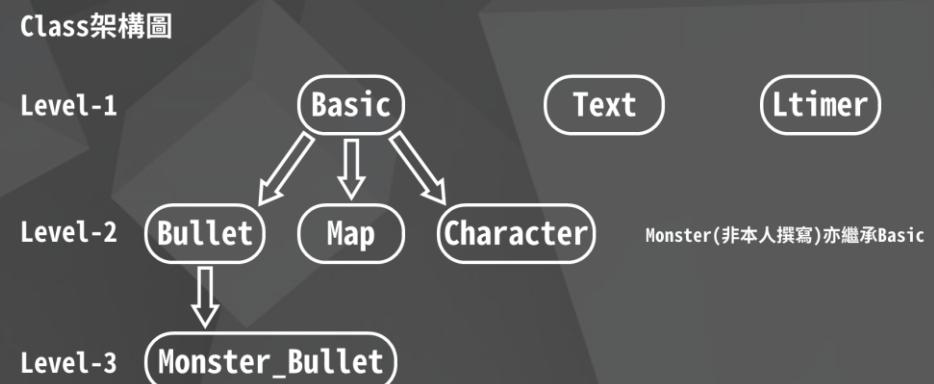
```
class LTimer
{
public:
    //Initializes variables
    LTimer();
    //The various clock actions
    void start();
    void stop();
    void pause();
    void unpause();
    //Gets the timer's tick count
    Uint32 getTicks();
    //Checks the status of the timer
    bool isStarted();
    bool isPaused();
private:
    //The clock time when the timer was started
    Uint32 mStartTicks;
    //The ticks stored when the timer was paused
    Uint32 mPausedTicks;
    //The timer status
    bool mPaused;
    bool mStarted;
};
```

# A.

程式撰寫前先制定整體架構，分析類別間需要的功能，找出共同項，思考設計彈性大且相容性高之基礎類別、便較容易使用繼承或組裝從一個基礎類別來擴充以及實作新的類別與功能，充分使用既有之程式碼。

封裝級別則大體上由Basic完成，諸如長寬、位置、Texture等不該被輕易更改之變數皆設定為private。

因多人撰寫專案，為避免混淆混用，故在專案中原先可能需設定之friend關係經過修改後則無設定。



## B.

**constructor**在各類別中，負責將變數初始化，亦有設定動態陣列。  
**destructor**則負責由**free**函式**deallocate**類別中的指標。

```
Character::Character()
{
    //Initialize
    Texture = NULL;
    Pos_x = 450;
    Pos_y = 450;
    Character_Hp = 100;
    Vel_x = 0;
    Vel_y = 0;
    Character_SpriteClips = new SDL_Rect[14];
}
Character::~Character()
{
    //DeAllocate
    free();
}

Map::Map()
{
    Pos_x = 0;
    Pos_y = 0;
}
Map::~Map()
{
    free();
}

Monster_Bullet::Monster_Bullet()
{
    Pos_x = 0;
    Pos_y = 800;
    Vel_x = 0;
    Vel_y = 0;
    damage = 5;
    speed = 6;
    bullet_bounce = false;
    bullet_type = 1;
}
void basic::free()
{
    if(Texture != NULL){
        SDL_DestroyTexture( Texture );
        Texture = NULL;
    }
}
```

## B.

除constructor外，因為特定類別需要隨關卡進行大幅度更改，  
亦有負責重新設定以及初始化之函式。

```
20 void Monster_Bullet::set_bullet(bool bounce, int spe, int dama, int type)
21 {
22     bullet_bounce = bounce;
23     speed = spe;
24     damage = dama;
25     bullet_type = type;
26
27     if(type==2) this->free(), this->loadFromFile( "./image\\bullet2.png" );
28     else if(type==3) this->loadFromFile( "./image\\bullet3.png" );
29     else if(type==5) this->loadFromFile( "./image\\bullet4.png" );
30
31 }
```

(Monster\_Bullet.cpp)

B.

負責材質載入初始化之函式(loadFromFile之覆寫與再利用)。

左函式亦包含設定動畫Clip需要之Rect，右函式則包含指定顏色去背功能。

```
36  bool Character::loadFromFile(std::string path, int w, int h)
37  {
38      bool ans = basic::loadFromFile(path);
39      Width /= w, Height /= h;
40      for( int i = 0; i < 8; i++ ){
41          Character_SpriteClips[ i ].x = (i*8)*Width;
42          Character_SpriteClips[ i ].y = (i/8)*Height;
43          Character_SpriteClips[ i ].w = Width;
44          Character_SpriteClips[ i ].h = Height;
45      }
46      for( int i = 8; i < 14; i++ ){
47          Character_SpriteClips[ i ].x = (i-8)%6*Width;
48          Character_SpriteClips[ i ].y = ((i-8)/6+1)*Height;
49          Character_SpriteClips[ i ].w = Width;
50          Character_SpriteClips[ i ].h = Height;
51      }
52      return ans;
53  }
```

(Character.cpp)

```
5  bool basic::loadFromFile( std::string path ,Uint8 R,Uint8 G ,Uint8 B)
6  {
7      //Get rid of preexisting texture
8      free();
9      //The final texture
10     SDL_Texture* newTexture = NULL;
11     //Load image at specified path
12     SDL_Surface* loadedSurface = IMG_Load( path.c_str() );
13     if( loadedSurface == NULL )
14     {
15         printf( "Unable to load image %s! SDL_image Error: %s\n", path.c_str(), IMG_GetError() );
16     }
17     else
18     {
19         //Color key image
20         SDL_SetColorKey( loadedSurface, SDL_TRUE, SDL_MapRGB( loadedSurface->format, R, G, B ) );
21
22         //Create texture from surface pixels
23         newTexture = SDL_CreateTextureFromSurface( gRenderer, loadedSurface );
24         if( newTexture == NULL )
25         {
26             printf( "Unable to create texture from %s! SDL Error: %s\n", path.c_str(), SDL_GetError() );
27         }
28         else
29         {
30             //Get image dimensions
31             Width = loadedSurface->w;
32             Height = loadedSurface->h;
33         }
34         //Get rid of old Loaded surface
35         SDL_FreeSurface( loadedSurface );
36     }
37     //Return success
38     Texture = newTexture;
39     return Texture != NULL;
40 }
```

(Basic.cpp)

# C.

## 1. virtual function

如前頁之例子，loadfromfile函式為virtual bool function，  
撰寫成virtual function 避免掉了程式碼的replication。  
也更好靈活運用開發過的程式碼。

```
8 class basic{
9     public:
10    virtual bool loadFromFile( std::string path, Uint8 = 0x00, Uint8 = 0x00, Uint8 = 0xFF );
11 class Character : public basic
12 {
13     public:
14         //constructor
15         Character();
16         //destructor
17         ~Character();
18         virtual bool loadFromFile(std::string path, int = 8, int = 2);
```

(Basic.cpp)

(Character.cpp)

# C.

## 1.virtual function

```
9  class Bullet : public basic
10 {
11     //friend Character;
12     public:
13         //Initializes variables
14         Bullet();
15         //Deallocates memory
16         ~Bullet();
17         //Loads image at specified path
18         virtual void set_v(int, int );
```

(Bullet.h)

```
class Monster_Bullet : public Bullet
{
    public:
        Monster_Bullet();
        virtual void set_v(int, int, int = 0, float = 1);
```

(Monster\_Bullet.h)

```
23 void Bullet::set_v( int des_x, int des_y){
24
25     static double dis,dx,dy;
26
27     Pos_x = gCharacter.getPos_x() + 30;
28     Pos_y = gCharacter.getPos_y() + 40;
29
30     dx = (des_x - Width/2 - Pos_x);
31     dy = (des_y - Height/2 - Pos_y);
32     dis = sqrt(dx*dx+dy*dy);
33     //std::cout<<"dis = "<<dis<<std::endl;
34     Vel_x = dx / dis;
35     Vel_y = dy / dis;
36     //std::cout<<"set_v"<<Vel_x<<' '<<Vel_y<<std::endl;
37 }
```

(Bullet.cpp)

```
47 void Monster_Bullet::set_v(int Mon_Pos_x, int Mon_Pos_y, int angle, float speed)
48 {
49     static double dis,dx,dy,theta;
50
51     Pos_x = Mon_Pos_x + 30;
52     Pos_y = Mon_Pos_y + 20;
53
54     dx = (gCharacter.getPos_x() + gCharacter.getWidth()/2 - Width/2 - Pos_x);
55     dy = (gCharacter.getPos_y() + gCharacter.getHeight()/2 - Height/2 - Pos_y);
56
57     theta = atan(dy/dx) ;
58     //std::cout<<"dis = "<<dis<<std::endl;
59     if(dx<0&&dy>0) theta+=3.14;
60     if(dx>0&&dy<0) theta-=3.14;
61     theta += 3.14*angle/180.0;
62
63     Vel_x = cos(theta)*speed;
64     Vel_y = sin(theta)*speed;
65 }
66 }
```

(Monster\_Bullet.cpp)

# C.

## 1. virtual function

如前頁之例子，set\_v函式為virtual void function，  
在Monster\_Bullet.cpp中的set\_v多出了調整偏向角與速度的功能，  
使怪物的子彈能夠有更多的樣式組合可能性(在D部分說明)。

```
23 void Bullet::set_v( int des_x, int des_y){  
24  
25     static double dis,dx,dy;  
26  
27     Pos_x = gCharacter.getPos_x() + 30;  
28     Pos_y = gCharacter.getPos_y() + 40;  
29  
30     dx = (des_x - Width/2 - Pos_x);  
31     dy = (des_y - Height/2 - Pos_y);  
32     dis = sqrt(dx*dx+dy*dy);  
33     //std::cout<<"dis = "<<dis<<std::endl;  
34     Vel_x = dx / dis;  
35     Vel_y = dy / dis;  
36     //std::cout<<"set_v"<<Vel_x<<' '<<Vel_y<<std::endl;  
37 }
```

(Bullet.cpp)

```
47 void Monster_Bullet::set_v(int Mon_Pos_x, int Mon_Pos_y, int angle, float speed)  
48 {  
49     static double dis,dx,dy,theta;  
50  
51     Pos_x = Mon_Pos_x + 30;  
52     Pos_y = Mon_Pos_y + 20;  
53  
54     dx = (gCharacter.getPos_x() + gCharacter.getWidth()/2 - Width/2 - Pos_x);  
55     dy = (gCharacter.getPos_y() + gCharacter.getHeight()/2 - Height/2 - Pos_y);  
56  
57     theta = atan(dy/dx) ;  
58     //std::cout<<"dis = "<<dis<<std::endl;  
59     if(dx<0&&dy>0) theta+=3.14;  
60     if(dx>0&&dy<0) theta-=3.14;  
61     theta += 3.14*angle/180.0;  
62  
63     Vel_x = cos(theta)*speed;  
64     Vel_y = sin(theta)*speed;  
65 }  
66 }
```

(Monster\_Bullet.cpp)

# C.

## 2. overloading

透過overloading Map類別之operator~來進行渲染，  
同時利用了原先繼承的Basic中的render函式來利用已開發之程式碼。

```
34 void Map::operator~ ()
35 {
36     //Set rendering space and render to screen
37     SDL_Rect renderQuad2 = { 0, 0, Width, Height };
38     if(stage_number%4 == 1) renderQuad2.y = 1600;
39     else if(stage_number%4 == 3) renderQuad2.y = 800;
40     basic::render(&renderQuad2);
41 }
42 void basic::render(SDL_Rect * clips)
43 {
44     SDL_Rect renderQuad = { Pos_x, Pos_y, Width, Height };
45     SDL_RenderCopy(gRenderer, Texture, clips, &renderQuad);
46 }
```

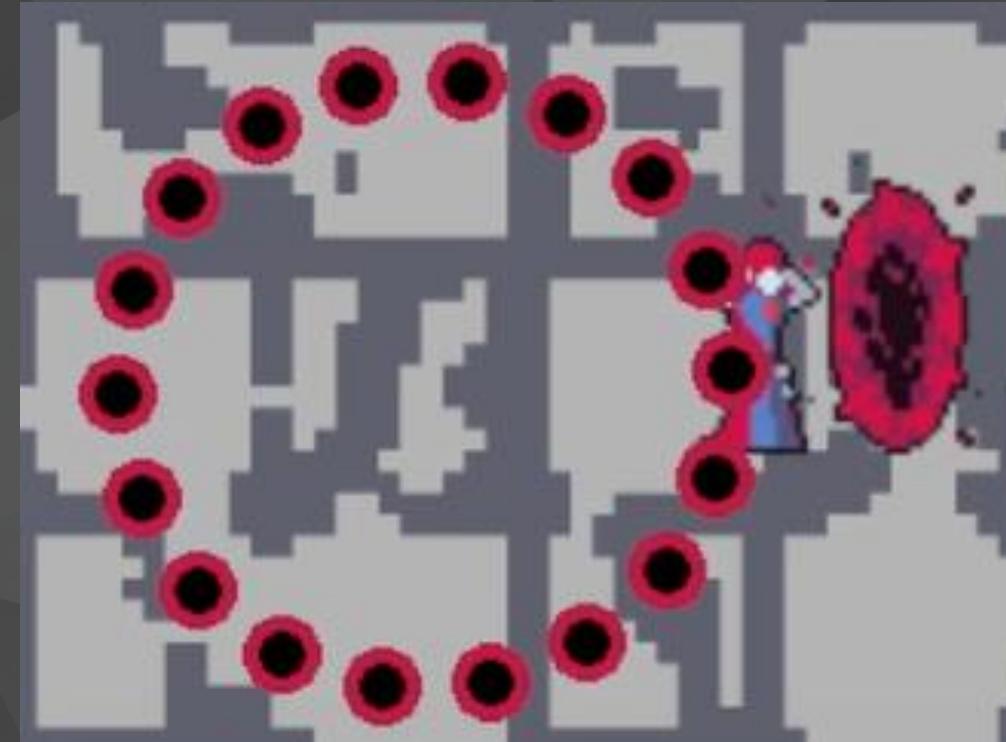
(Map.cpp)

(Basic.cpp)

# D.

## 1. 怪物子彈設計

遊戲的三個魔王有著不同的子彈射擊模式，其中經過了一番思考琢磨。



D.

## 1. 怪物子彈設計

魔王3的子彈具有追蹤或反彈的能力(*Monster\_Bullet.cpp*)

追蹤部分在一定距離內會停止並加快飛行速度，要防止二次追蹤也需要經過調整。

```
129 void Monster_Bullet::change_v()
130 {
131     if(!track) return;
132     static double dx,dy,theta;
133     if(Pos_y!=800){
134         dx = (gCharacter.getPos_x() + gCharacter.getWidth()/2 - Width/2 - Pos_x);
135         dy = (gCharacter.getPos_y() + gCharacter.getHeight()/2 - Height/2 - Pos_y);
136         theta = atan(dy/dx) ;
137         if(dx<0&&dy>0) theta+=3.14;
138         if(dx<0&&dy<0) theta-=3.14;
139     }
140     if(sqrt(dx*dx+dy*dy)>150){
141         //std::cout<<"endkp"<<abs(Vel_x)<<abs(Vel_y)<<std::endl;
142         Vel_x = cos(theta)*0.4;
143         Vel_y = sin(theta)*0.4;
144     }
145     if (sqrt(dx*dx+dy*dy)<150){
146         Vel_x = cos(theta);
147         Vel_y = sin(theta);
148         track = false;
149     }
150 }
```

反彈則在檢查邊界的函式進行修改。

```
79 void Monster_Bullet::check_edge()
80 {
81     static int ct = 0;
82     if(bullet_bounce){
83         Bullet::check_edge();
84     }else{
85         int ans = 0;
86         if(stage_number%2 == 0){
87             if(Pos_x < 450) ans = true;
88             else if(Pos_x > 950) ans = true;
89             if(Pos_y < 100) ans = 2;
90             else if(Pos_y > 600) ans = 2;
91         }else if(stage_number%2 == 1){
92             if(Pos_x > 950) ans = true;
93             if(Pos_y > 600) ans = 2;
94             if(Pos_x < 200) ans = true;
95         }
96         if(ans){
97             if(ct<2){
98                 if(ans==1) Vel_x *= -1;
99                 if(ans==2) Vel_y *= -1;
100                 ct++;
101             }else{
102                 Vel_x = 0, Vel_y = 0;
103                 Pos_x = 0, Pos_y = 800;
104                 ct = 0;
105             }
106         }
107     }
108 }
```

# D.

## 2. 怪物設定

怪物的數量會隨關卡而更動，因此怪物和怪物子彈皆使用動態陣列，而關卡間的轉換過程，則包含了重新設定怪物與怪物子彈的函式(Next.cpp)

參數(X, Y, 行動模式, 種類, 血量)

```
48 void Create_Monster(int Monster_number, int stage_number)
49 {
50     delete [] gMonster;
51     gMonster = new Monster [Monster_number];
52
53     if(stage_number==1){
54         gMonster[0].set_Monster(300,500,1,1,15);
55         gMonster[1].set_Monster(400,400,2,1,15);
56         gMonster[2].set_Monster(700,200,3,1,15);
57     }else if(stage_number==3){
58         gMonster[0].set_Monster(300,500,1,1,15);
59         gMonster[1].set_Monster(400,400,2,1,15);
60         gMonster[2].set_Monster(700,300,3,1,15);
61         gMonster[3].set_Monster(800,250,2,1,15);
62     }else if(stage_number==5){
63         gMonster[0].set_Monster(400,400,4,4,50);
64     }else if(stage_number==7){
65         gMonster[0].set_Monster(300,500,1,2,20);
66         gMonster[1].set_Monster(400,300,2,2,20);
67         gMonster[2].set_Monster(700,400,3,2,20);
68     }else if(stage_number==9){
69         gMonster[0].set_Monster(300,500,1,2,20);
70         gMonster[1].set_Monster(400,400,2,2,20);
71         gMonster[2].set_Monster(700,300,3,2,20);
72         gMonster[3].set_Monster(800,250,2,2,20);}
```

參數(反彈與否, 速度, 傷害, 樣式)

```
92 //set_bullet(bool bounce, int spe, int dama, int type);
93 if(stage_number==1){
94     for(int i=0; i<Mon_Bullet_number ;i++)
95         mBullet[i].set_bullet(0,5,3,1);
96 }else if(stage_number==5){
97     for(int i=0; i<Mon_Bullet_number ;i++)
98         mBullet[i].set_bullet(0,8,5,2);
99 }else if(stage_number==7){
100    for(int i=0; i<Mon_Bullet_number ;i++)
101        mBullet[i].set_bullet(0,6,4,3);
102 }else if(stage_number==11){
103    for(int i=0; i<Mon_Bullet_number ;i++)
104        mBullet[i].set_bullet(0,9,6,4);
105 }else if(stage_number==13){
106    for(int i=0; i<Mon_Bullet_number ;i++)
107        mBullet[i].set_bullet(0,7,5,5);
108 }else if(stage_number==17){
109    for(int i=0; i<Mon_Bullet_number ;i++)
110        mBullet[i].set_bullet(1,9,7,6);
111 }
112 }
113 }
```

D.

### 3. 魔王召喚小怪

魔王2具有召喚小怪的能力，但為了模擬出召喚，進行了以下工程。



製作並潤飾召喚的動畫



進入關卡前，預先設定被召喚小怪為死亡狀態，並設定為至於視窗外等待被召喚加入戰局。(Next.cpp)

```
73 }else if(stage_number==11){  
74     gMonster[0].set_Monster(0,800,7,7,20);  
75     gMonster[1].set_Monster(0,800,7,7,20);  
76     gMonster[2].set_Monster(0,800,7,7,20);  
77     gMonster[3].set_Monster(0,800,7,7,20);  
78     gMonster[4].set_Monster(400,400,5,5,60);
```

D.

#### 4.受擊回饋

在玩家遭受子彈攻擊時，不只有音效提示，  
在顯示面板上的玩家生命值也會有對應的閃爍功能。

(main.cpp)

```
115 //Set Text
116 Hp_num.str("");
117 Hp_num<<"HP : "<<gCharacter.get_Hp();
118 if(Hurt&&timer%8<4) Text_Hp.loadFromRenderedText( Hp_num.str().c_str(), Text_Hp.setColor(255,0,0) );
119 else Text_Hp.loadFromRenderedText( Hp_num.str().c_str(), Text_Hp.setColor(255,255,255) );
120 if(timer==timesave) Hurt = false;
```



D.

## 5.多組動畫處理方式

主角具有多組動畫(走動、攻擊)，而停滯時不會顯示走動動畫，



但此時若攻擊則依然會顯示攻擊動畫，  
因此需對不同情況動畫的優先度進行處理。

(Character.cpp)

```
55 void Character::render()
56 {
57
58     //Set rendering space and render to screen
59     //scale = 0.7
60     SDL_Rect renderQuad = { Pos_x, Pos_y, Width, Height };
61     //Character_SpriteClips[ mCurrentSprite ] control the sprite will show Later
62     //第三個RECT跟圖片本身攝取有關第四個RECT跟在螢幕上的哪裡有關
63     int sprite = atk*(8+attack_timer/7)+(!atk)*(+attack_timer/5)*(!stp);
64     if(lr) SDL_RenderCopyEx(gRenderer, Texture, &Character_SpriteClips[sprite], &renderQuad,0,NULL,SDL_FLIP_HORIZONTAL);
65     else   SDL_RenderCopy(gRenderer, Texture, &Character_SpriteClips[sprite], &renderQuad);
66
67     if(attack_timer>36) atk = 0;
68
69 }
```

其中sprite控制動畫的Clip，  
atk與stp則是記錄當前是否攻擊與停滯。

期末報告結束

謝謝您的閱讀