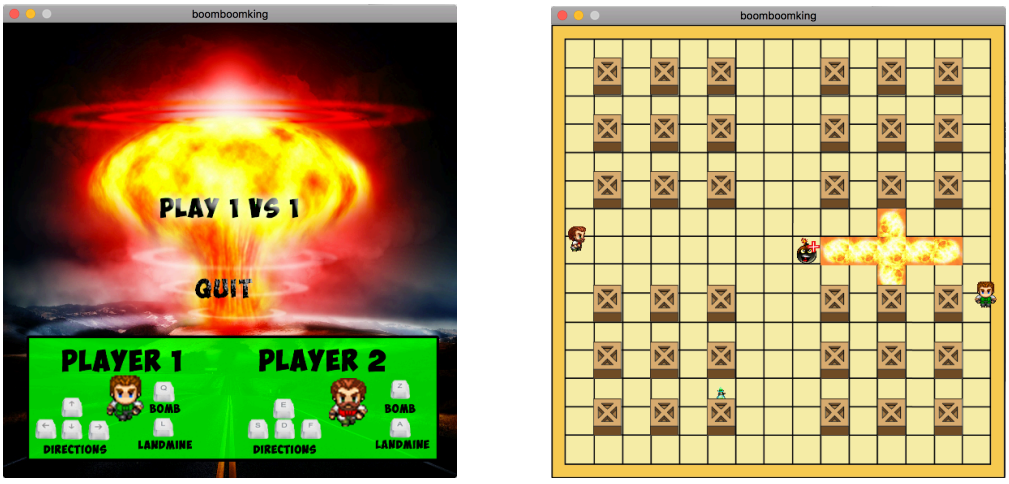
BoomBoomKing

陳奐廷

January 15, 2018

0 遊戲簡介

BoomBoomKing 這款遊戲，是以知名遊戲「爆爆王」為藍圖，再增加一些功能，綜合所得的一款兩個玩家對戰的遊戲。遊戲的目標是把敵方炸死。每個玩家可以下炸彈（不限顆數），以及地雷（限兩顆）。地圖上一開始會有道具，之後每隔一段時間會產生新的道具。道具有兩個：超級炸彈以及無敵藥水。當你吃到超級炸彈，你的炸彈所爆炸的範圍會變大；而無敵藥水的效果是不會被炸彈所炸死，但仍然無法逃過地雷。



(a) main menu

(b) while playing

Figure 1: 遊戲介面

1 程式分工

我們的專題使用了大量的 class 來實作，每個組員分別寫幾個不同的 class，而最後是由我來彙整所有的檔案，再做一些必要的修改。這樣的好處是我的 main.cpp 檔裡面非常的乾淨，只有少數幾個函式而已，而且日後如果需要擴充功能，或是需要做一些調整都非常的方便。在這次的專題裡，我貢獻的檔案有 main.cpp, background.cpp, tools.cpp, landmine.cpp, show_bomb.cpp, LTexture.cpp, 部分的 characher.cpp（還有這些檔案相對應的.hpp 檔）。

2 Class 實作

以下舉的例子是我所寫的 code 裡，我覺得值得特別拿出來討論的部分。

2.1 LTexture

將 Lazyfoo sample code 輸入圖檔的部分另外寫成一個 class，再稍作修改，使之符合我們需要。在解構的部分，我們把解構的函式另外寫成一個函式 `free()`，方便之後使用。

```
1  class LTexture{
2  public:
3      LTexture();  //constructor
4      ~LTexture(); //destructor
5      void free(); //deallocate
           function
6      //omit some functions here
7  private:
8      //omit some variable
9      SDL_Texture* mTexture;
10     int mWidth;
11     int mHeight;
12 };
```

Listing 1: LTexture.hpp 18-49

```
1  LTexture::~LTexture(){
2      free();
3  }
4  void LTexture::free(){
5      if( mTexture != NULL )
6      {
7          SDL_DestroyTexture(
               mTexture );
8          mTexture = NULL;
9          mWidth = 0;
10         mHeight = 0;
11     }
12 }
```

Listing 2: LTexture.cpp 23-27,71-81

2.2 Background Class

background class 是為了輸出背景及障礙物所寫的 class。包含了所有與輸出背景有關的函式，用 `box_rerendering` 來解決人物與障礙物重疊的問題。
有寫 constructor 來初始化數值，不過在這個 class 沒有用到任何的指標，所以就不需要再另外寫 destructor，直接用預設的就可以了。

```
1  class Background{
2  private:
3          //omit some variables here
4  public:
5      Background(); //constructor
6      bool loadMedia(); //loadMedia function
7      void background_rendering(); //render the background
8      void box_rendering(); //render the box
9      void box_rerendering(int, int); //rerender boxes that overlap with the
           character
10     bool check_character_behind_box(int, int);
11     SDL_Rect blockwall[15][15];  //————blockwall[y-axis][x-axis]
12 };
```

Listing 3: background.hpp 18-34

2.3 Bomb Class

Bomb Class 是我認為封裝的最完整的一個 class。

1. 有 privite,public 的區分，而且將所有與炸彈有關的函式全部寫進來，相當的完整。

2. 在 `set_bomb_position` 這個函式當中，以 call by reference 的方式，讓位置不會跑掉。

3. 在這個 class 中，沒有用 destructor，取而代之的是寫一個解構的函式。是因為在主程式中，這個 class 是重複使用的，我不希望在執行結束之後馬上把這個 class 解構掉，所以在主程式的最後才呼叫 `free()` 把它解構掉。(其實就是因為加了 destructor 程式會有 bug，經過無限的 debug 之後，發現把 destructor 拿掉就過了...)

```
1  class Bomb{
2  private:
3      //omit some variables and pointers here
4  public:
5      Bomb();             //constructor
6      void free();        //deallocate function
7      enum bomb_index_number{  /*omit the enumeration*/  };
8      enum KeyPressSurfaces{  /*omit the enumeration*/  };
9      void set_bomb_position(double &,double &);
10     void bomb_display(int, int);                //display bombs on the map
11     SDL_Texture* texture_bomb;      //bomb texture declaration
```

Listing 4: show_bomb1.h 15-88

2.4 Character Class

這個 class 中有一部分不是我主筆，但是有一大部分是我主筆的，另一部份我也有參與討論，故也放上來。

1. 在 constructor 的部分，因為要針對兩個不同的角色做不同的初始化，所以用 `charachter(enum)` 的方式來寫。雖然只有兩個角色，但是用 enum 寫的好處就是之後如果要新增更多角色，這樣再去更改會比較容易和美觀。

2. 在參數的設定上，有加上 `const`, `static`，這樣避免其他的函式不小心去修改到他的數值。其中有一個參數忘記加 `const` 雖然不會有問題但是我覺得還是不好。

3. `check_Collision(SDL_Rect, SDL_Rect*)` 是為了檢查角色有沒有撞到障礙物所寫的函式。因為障礙物是用 15*15 的陣列去存，所以我讓他傳入一個指標，就可以用一個兩層的迴圈，然後裡面放 `temp[i][j] = *(b+15*i+j);`。

4. 我有重新定義一個 operator= ，因為在建立 contructor 的時候，同樣 argument 數的建構式有不只一個，所以他的 operator= 會不能用，需要重新定義成 default。不過後來主程式需要用到這個的部分改掉了，所以我把這個 operator 註解掉了。

```
1  enum player{  /*omit the enumeration*/  };
2  class Charachter{
3  private:
4      //omit some variables and pointers here
5      const int WALKING_ANIMATION_FRAMES = 3; //parameter setting
6      int WALKING_ANIMATION_SIZE = 32;    //parameter setting (adding 'const'
           in the front should be better but i forgot orz)
7  public:
8      static const int CHAR_WIDTH = 32;   //parameter setting
9      static const int CHAR_HEIGHT = 32;  //parameter setting
10     static const double CHAR_VEL = 1;   //parameter setting
11     Charachter(enum player playable);   //different constructors for each
           character(player)
12     bool checkCollision( SDL_Rect, SDL_Rect* );    //check collision
           between character and obstacles
13     Charachter& operator= (const Charachter& a) = default; //this line is
           commented in the file
14 };
```

Listing 5: Charachter.hpp 20-105

3 其他程式細節

以下介紹一些主程式碼中，我覺得還蠻有巧思的地方，以及我 debug 很久的地方。

3.1 Thread

因為程式的需要，我必須讓主程式運作的同時，也做其他事情。例如說炸彈引爆的連續動作中，角色必須繼續移動。所以我需要同時平行的去執行一些程式碼，也就是利用 Thread 來增加執行序。C++ 中有內建的 <thread> 可以用，不過在專題裡，我是使用 SDL Library 裡的 thread 功能。

3.2 Bomb Declaration

在遊戲裡，玩家可以放置炸彈的數量是無上限的，而每一顆炸彈都是一個新的 class，所以我開了一個陣列 `bomb_array[MAX_BOMB]` 來存每一個 bomb class。`MAX_BOMB` 是同時可以存在的炸彈個數，我用 define 去定義它對於之後要改參數會比較方便。當數量超過 `bomb_array[MAX_BOMB]` 時，會把第一顆取代掉。同理也有一個 `bomb_showing_threads[MAX_BOMB]` 讓每一個 bomb 都有一個 thread。這應該也是造成在2.3節中第3點裡提到的加上 destructor 會有 bug 的原因：我有可能還會重複使用第一個 bomb class，所以當第一個 bomb class 執行完，我仍然不能解構它，要保留他的位置。

有個值得一提的點是，我一開始並不是用 array 去存 bomb class，我是用 vector 去存。我認為 vector 能夠自由伸縮長度，每多下一顆炸彈，就可以用 push_back() 加在這個 vector 後面，比 array 自由許多。但是結果是出現很奇怪的 bug：有一些炸彈很順利的執行，而有一些會有問題。debug 很久之後才發現，問題出在 vector 運作的原理上面。vector 會先給定一個預設大小的記憶體，當他已經滿了的時候，在新增一個物件進去，它會把整個 vector 搬到另一個地方，並且空間是前一個的兩倍大。當我某一個炸彈的 thread 做到一半，剛好進行這個搬移的動作，那存在原本那個位址的變數就整個跑掉導致有 bug。

```
1  #define MAX_BOMB 50
2  Bomb bomb_array[MAX_BOMB]={};   //bombs array (allowing to show MAX_BOMB
       bombs at the same time)
3  SDL_Thread* bomb_showing_threads[MAX_BOMB];   //threads array
4  if(e.key.keysym.sym==SDLK_l){
5      double x = charachter1.getposx(), y=charachter1.getposy();
6      Bomb a;
7      a.bomb_character=1;
8      if( !a.loadMedia() ) { printf( "Failed to load bombs media!\n" );}
9      else{
10         bomb_array[count % MAX_BOMB] = a;
11         bomb_array[count % MAX_BOMB].set_bomb_position(x,y);
12         bomb_showing_threads[count % MAX_BOMB] = SDL_CreateThread(
               show_bomb_thread, "show_bomb",(void*)(bomb_array+(count %
               MAX_BOMB)));   //creating thread for each bomb
13         count++;
14     }
15 }
```

Listing 6: main.cpp 71-75, 234-245

3.3 Close

在主程式的最後有一個 close() 函式。裡面充斥著 free() 函式（之前沒有把它解構掉的東西現在才來解構）和一些 SDL 相關的關閉視窗，讓程式順利結束。

```
1  void close(){
2      for(int i=0;i<MAX_BOMB;i++){
3          bomb_array[i].free();
4      }
5      SDL_DestroyRenderer( gRenderer );
6      gRenderer = NULL;
7      SDL_DestroyWindow( gWindow );  //Destroy
           window
8      gWindow = NULL;
9      IMG_Quit();    //Quit SDL subsystems
10     SDL_Quit();
11 }
```

Listing 7: main.cpp 576-610

4 奇聞軼事

多人一起在寫程式的時候，往往會出現一些 bug。有一些 bug 雖然需要 de 很久，但是至少是已知的 bug。而有一些真的就非常的神奇，至今仍然未解。

4.1 神奇的 13 月

我的 Mac 是 High Sierra 最新版，Xcode 也有更新到最新，但是只要用到 SDL Library，就會有 bug。編譯會過，但是執行出來第一行會顯示：Month 13 is out of bounds，然後就無法繼續執行下去。所以我只好改開 terminal 用 g++ 編譯，就可以正確執行。我猜測是跟之前 Apple 出大 bug（iphone 時間壞掉，全部當機）一樣的原因，可是到現在仍然不能用。

4.2 神奇的 Mac&Windows

Mac 和 Windows 的作業系統不知道為什麼也會有神奇的差異。

4.2.1 神奇的 const

剛開始做專題就發現了這個問題：所有在 Mac 上的 code 丟到 windows 去都不能編譯。後來才發現是因為 Xcode 的預設檔案裡，是用 `int main()(int argc, const char * argv[])`，然後要把 `const` 刪掉，才可以在 windows 系統順利執行。

4.2.2 神奇的編譯結果

全部寫完專題的時候發現，用一模一樣的 code，但是在兩個系統底下會產生不一樣的結果。

1. 在 Mac 底下完全可以正常運作的 code，用 windows 下會停止運作。問題出在於第3.3節講到的 `close()` 函式。在專題截止死線前夕，我才忽然發現在 windows 系統底下會出事，所以只好把那段函式註解掉一部份，才可以順利執行。

2. 在兩個系統底下執行出來的結果不一樣。在人物死掉的時候，在 Mac 系統底下，爆炸火花會閃，如 demo 影片裡大約 4:24-4:29, 4:40-4:45 等片段所拍的。參見（www.youtube.com/watch?v=XMvianJXzWo）。可是在 windows 系統底下跑，卻不會有這樣的現象發生。