B07901033 施貽仁

負責部分：網路連線，登入畫面，主選單，遊戲關卡介面，角色選取畫面，背景音樂製作

以下大約分幾點介紹(由於逐行說明篇幅偏大，所以我用以下幾個大方向說明)

1. 概述 各 class 功能 and 架構
2. Render 機制說明 (畫面方面)
3. Callback 機制說明 (互動方面)
4. 巧思-Scroll List
5. 網路連線說明
6. 遊戲流程控制
7. 心得

Class 功能 and 架構：

概念：每個出現的畫面都實作為 class Form 的 derived class

每個畫面上的控制項實作為 class GameObj 的 derived class

（我會想到用這個方法主要是和以前使用 vb.net 建立 Window Form 時所啟發）

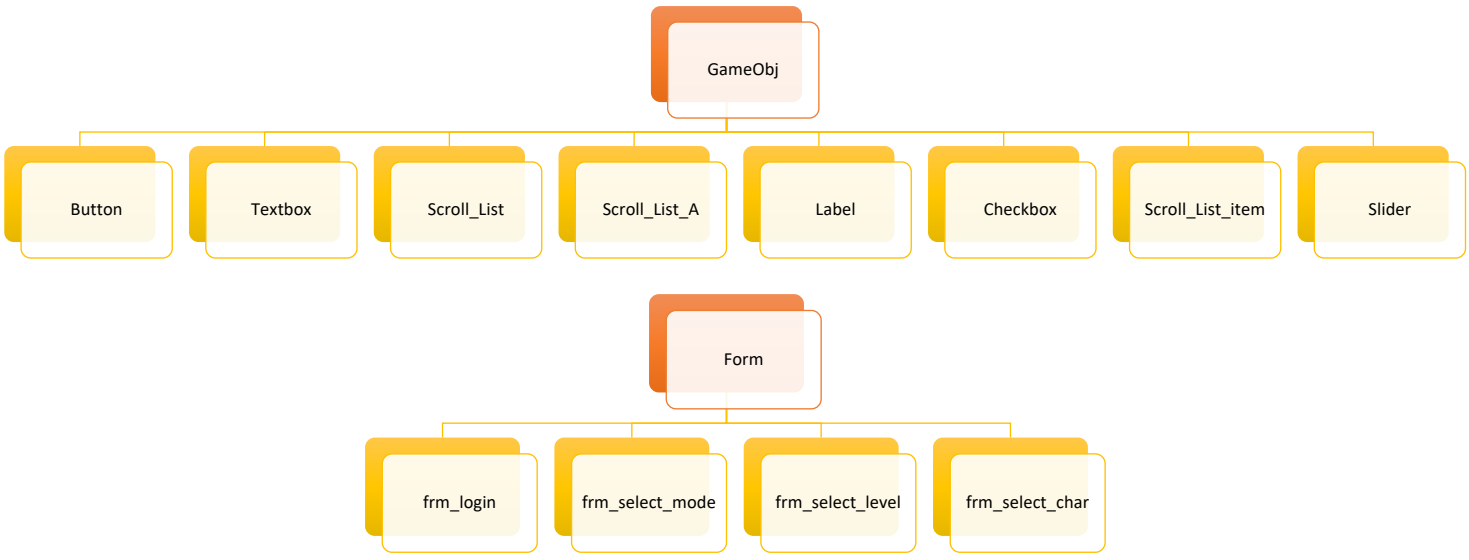
再設計一個 class Game 用來儲存遊戲資料，玩家 ID

Class LTexture (LTexture.h) 是從 LazyFoo 的範例略為修改

Class ServerRequest (ServerRequest.h) 用來使用 socket 從我家主機下載資料

Class List (myqueue.h) 自己設計的結構，處理滾動選單部分

以下是 class 的繼承關係



接著我以兩個主要觀點來說明我的設計

遊戲個物件 render 的角度

各物件 event callback

先講 物件的 render

Render 機制：

從 base class GameObj (GameObj.h)講起

```
12  #include "LTexture.h"
13  #include "Game.h"
14
15  class GameObj
16  {
17  public:
18
19      GameObj(Game*,void* =0);
20      GameObj();
21      ~GameObj();
22
23      void assign(Game*,void* =0);
```

Constructor 的部分：

1.有兩個參數(Game，和一個 void pointer)

Game 得知遊戲的畫面長寬，此物件才能 render

void pointer 主要是為了 event callback 之後詳述

2.沒有參數的 Constructor 以便程式可以用 new 的方式產生物件

3. assign 函數配合 new 的方式，在物件產生後再把參數指派給物件

```
26      void setX(int a);
27      void setY(int a);
28      void setPosition(int a,int b);
29      SDL_Point mPosition;
30
31      bool setObjWidth(int);
32      bool setObjHeight(int);
33      int getObjWidth() const;
34      int getObjHeight() const;
```

設定，取得物件 x,y 長寬的函數

```
36      void setOpacity(Uint8);
37      bool loadBG(std::string); //do not use it
38      bool loadCaption(std::string font_path,int font_size,std::string text,SDL_Color& c) ;
39
40      void setBg(LTexture *,SDL_Rect* =0);
41      void setFore(LTexture *,SDL_Rect* = 0);
42
43      void resetFore();
44      void resetBg();
45
46      void setFocus(bool = 1);
47      void setFocusDetectFlag(bool = 1)
48      bool getFocusDetectFlag() const;
49      bool isFocus() const;
50      void setNextObj(GameObj* =0);
51      GameObj* getNextObj() const;
```

設定可見度

載入背景（後在沒用，為省記憶體）

載入文字

將背景，前景指定到給定的 Texture

將背景，前景指標歸零

這部分主要是處理使用者按下 tab 鍵切換控制項

當物件 isFocus 時，鍵盤輸入的東西會被物件記錄下來

DetectFlag 是為了給程式緩衝時間，在偵測 tab 鍵

NextObj 是當此物件 isFocus 時按下 tab 鍵所跳到的控制項

```
55      virtual void render();
56      void renderBG();
57      void renderFont();
```

每個 GameObj 都可以 render 但他們各自的 render 可能不同，所以我用 virtual 當我用 base class pointer 可指到各自的 function

Render 裡執行 renderBG 和 renderFont

以上是有關 render 部分的 public member 這些功能都是可被外面呼叫

接下來是 protected 部分

```
protected:

    Game* g;
    void *form_ptr;

    LTexture mfont_texture;

    LTexture* bg_texture=0;
    SDL_Rect* bg_texture_rect=0;
    LTexture* font_texture=0;
    SDL_Rect* font_texture_rect=0;
```

這些 member 只需讓繼承的 GameObj 使用即可

外面不該可以更動

儲存背景，前景 Texture 指標

接下來是衍生物件類別

以 button 為例 (Button.h)

```
#ifndef BUTTON_H
#define BUTTON_H

#include "GameObj.h"

class Button : public GameObj {
public:
    //void render();
    Button(Game*,void* =0);
    Button();
    void handleEvent( SDL_Event* e );


private:


};
Button::Button(Game *x,void * z):GameObj(x,z) {}
void Button::handleEvent(SDL_Event *e)
{
    GameObj::handleEvent(e);
}

Button::Button() {

}
```

用 public 方式繼承 GameObj

Button 由於 render 部分沒有特殊，故直接利用 base class 的 render 即可

再以 Textbox 為例(Textbox.h)

```
#include "GameObj.h"
#include <iostream>
class Textbox: public GameObj
{
public:
    //void render();
    Textbox(Game *,void * =0);
    ~Textbox();
    void handleEver
    void render();
```

用 public 方式繼承 GameObj

可使用 protect 部份

重新定義 render 由於 Textbox 的 render 要考慮鍵盤的輸入

```
void Textbox::render() {
    this->renderBG();
```

Line104

在 render 部分由於背景的 render 方式和 base class 一樣

所以直接呼叫 base class renderBG，下方(這邊沒有截圖)在加入 Textbox 特殊的 render 需求

直得留意的是 render 的先後影響到圖層的概念，越後面 render 會蓋住前面，所以背景先 render

以 Scroll_List(Scroll_List.h)為例：

```
#include "Scroll_List_item.h"
#include "myqueue.h"
#include <iostream>
#include <vector>
class Scroll_List: public GameObj
{
public:
    Scroll_List(Game*,void* =0);
    ~Scroll_List();
    void addButton(std::string,int=-1,bool isEnable=true,bool =
        0);
    void handleEvent( SDL_Event* e );
    void move();
    List<Scroll_List_item> listOfbuttons;
    void render();
```

用 public 方式繼承 GameObj

可使用 protect 部份

Scroll_List 裡的每一個選項為 Scroll_List_item 同屬於 GameObj

```
void Scroll_List::render() {

    GameObj::render();
    move();
    listOfbuttons.get_iterator(0);
    Scroll_List_item *a;
    while((a = listOfbuttons.get_iterator())!=NULL)
    {
        //std::cout<<a->mPosition.y<<" ";
        if(a->mPosition.y<this->mPosition.y)
        {

        }else if(a->mPosition.y>this->obj_HEIGHT+this-
            >mPosition.y)
        {

        }else{
            a->render();

        }
    }
```

重新定義 render

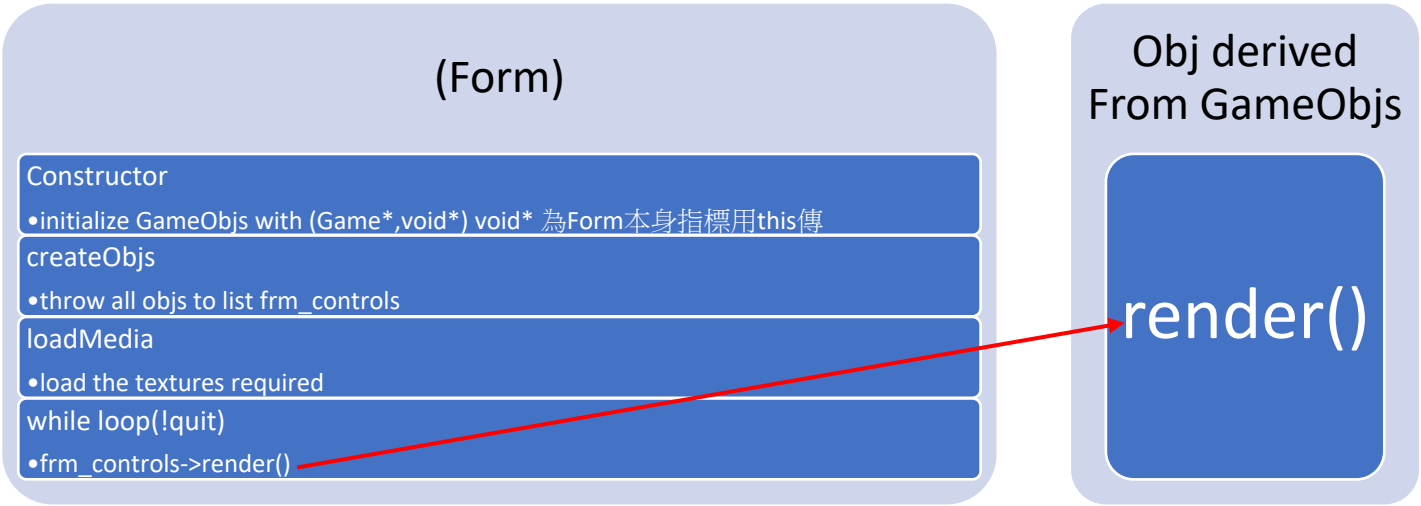
由於本身(Scroll_List)的 render 方法和 base class 同，直接先呼叫他

直接呼叫 Scroll_List_item 的 render，把各個 item render 出來

後方的程式碼主要是依照各選項的位置輸出，超過邊界者不顯示

在 Form 中 render 物件

大約結構



以下用 frm_select_char.h 說明：

先看 base class Form.h

```
#ifndef FORM_H
#define FORM_H

#include "Game.h"
#include "sdl_headers.h"

#include <stdio.h>
#include <string>
#include "LTexture.h"

#include "Button.h"
#include "Scroll_List.h"
#include "Scroll_List_A.h"
#include <iostream>
#include "myqueue.h"
#include "Textbox.h"
#include "Label.h"
#include "Slider.h"
#include "Checkbox.h"
#include <vector>
```

Form 中會用到的控制項先 include

```
class Form {
public:
    Form(Game*);
    ~Form();
    Game* g;
    void close();
    virtual bool createObjs()=0;
    virtual bool loadMedia()=0;
    void gotoFocus(GameObj*);
```

Constructor 也需要 Game

把 createObjs() 和 loadMedia()先定義，將 Form 變為 Abstract

這些 member 只需要由各個 form 去操作，故設為 protect

```
protected:
    bool quit = false;
    bool show_err=false;
    std::vector<GameObj*> frm_controls;
    GameObj* current_focus=0;
    bool detectForTab=true;
    std::vector<LTexture*> frm_textures;
    LTexture frm_err_msg;
    LTexture frm_background;
    void setFormBg(std::string path);
    void renderFormBg();
    void setFormErr(std::string path);
    void renderFormErr();
};
```

用來裝 form 上所有的控制項

由於要可以裝 Button，Textbox…

所以我用 Base class pointer 資料型態

再來看 frm_select_char.h

```
class frm_select_char: public Form {
private:
    int returnCmd = -1;

    Label lbl_title;
    Label lbl_info;
    Label lbl_level_bg;
    Scroll_List_A sl;
    Button btn_next_pg;
    Button btn_prev_pg;
    Slider sli_speed;
    Checkbox ch;

    bool createObjs();
    bool loadMedia();

    LTexture gTexture_item_bg;
    LTexture gTexture1_item_bg_selected;

    LTexture gTexture_left_arrow;
    LTexture gTexture_right_arrow;
    LTexture gTexture_level_bg;
    LTexture gTexture_slider_bg;
    LTexture gTexture_ch_bg;
```

以 public 方式繼承，方便取用 member

宣告這個 Form 中將使用的控制項

Base class 以先宣告，每個 derived class 都要

這個 Form 會用到的 Texture

Constructor:

```
frm_select_char::frm_select_char(Game *s) :
    Form(s),lbl_title(s),sl(s,this),ch(s,this),btn_next_pg(s,this)
    ,btn_prev_pg(s,this) ,sli_speed(s,this),lbl_info(s),lbl_level_
    bg(s){

}
```

把 Game 和 form pointer (有需要 call back 才需要)傳入個個控制項

```
bool frm_select_char::createObjs() {
    setFormBg("img/bar_bg1.png");
    int q = (g->SCREEN_WIDTH-330)/2;
    int u= 250;
    SDL_Color textColor = { 0, 0, 0, 0xFF };

    lbl_title.setPosition((g->SCREEN_WIDTH-300)/2,g->SCREEN_HEIGHT-75);
    lbl_title.setObjHeight(50);
    lbl_title.setObjWidth(300);
    lbl_title.loadCaption("16_true_type_fonts/GenSenMaruGothicTW-
        Regular.ttf",25,"Select game level",textColor);
    frm_controls.push_back(&lbl_title);
```

以建立一個 Label 說明

設定位置長寬

設定文字

將此物件的指標丟入 frm_controls

每次設定好一個控制項就把他丟到 frm_controls

```
bool frm_select_char::show(bool needPrevButton) {
    createObjs();
    //Load media

    if( !loadMedia() )
    {
        printf( "Failed to load media!\n" );
    }
    else
    {
        //Main loop flag


        //Event handler
        SDL_Event e;


        //While application is running
        while( !quit )
        {
```

show 是外部呼叫 from 時顯示的地方

先呼叫物件的設定

載入 texture

進入主迴圈，當 quit=1 迴圈結束，form 關閉

```
renderFormBg();
//Render buttons
for(int i=0;i<frm_controls.size();i++)
    frm_controls[i]->render();
if(needPrevButton)
    btn_prev_pg.render();

if(show_err)
    renderFormErr();
```

先 render 整個畫面的背景

逐一把物件 render

是否顯示錯誤訊息(壓在最上面)

PS:每個畫面的圖層結構：

(上面會蓋住下面)

Callback 機制：

用登入按鈕在 frm_Login.h 為例

設計方式：

在 class GameObj 裡(Game.h)

每個 GameObj 都有一些比較常見的事件類型在 base class 先定義

```
virtual void handleEvent( SDL_Event* e );
```

先宣告一種型態，讓下方的 set 函數可當成 parameter

```
typedef void (*event_callback_function)(void *,void*);
void setEventHandler_MousePressed(event_callback_function);
void setEventHandler_MouseRelease(event_callback_function);
void setEventHandler_MouseEnter(event_callback_function);
void setEventHandler_MouseLeave(event_callback_function);
void setEventHandler_KeyDown(event_callback_function);
```

把這些函數放在 public 使得每個 form(GameObj 外部)可使用

以下是 protected

```
void (*event_MousePressed)(void* o,void* frmptr)=0;
void (*event_MouseRelease)(void* o,void* frmptr)=0;
void (*event_MouseEnter)(void* o,void* frmptr)=0;
void (*event_MouseLeave)(void* o,void* frmptr)=0;
void (*event_KeyDown)(void* o,void* frmptr)=0;
```

在 protected 裡宣告各個 function pointer

兩個參數

前面的是目前 GameObj 物件的指標

後面的是控制向所在 form 的指標

```
void *form_ptr;
```

用來儲存此 GameObj 所屬的 form

在 GameObj:: handleEvent 裡我寫了一滑鼠事件基本的操作，因為很多控制項都會用，所以我寫在 base class 裡，而如果 derived class 物件有自己 handleEvent 的方式可以 override 或使用 base class 在修改

以下是節錄：(GameObj.h handevent())

```
if( !inside )
{
    if(event_MouseLeave!=NULL)
        event_MouseLeave(this,form_ptr);

}
    //Mouse is inside button
else
{
    //Set mouse over sprite
    switch( e->type )
    {
        case SDL_MOUSEMOTION:
            if(event_MouseEnter!=NULL)
                event_MouseEnter(this,form_ptr);
            break;

        case SDL_MOUSEBUTTONDOWN:
            if(event_MousePressed!=NULL)
                event_MousePressed(this,form_ptr);
            break;

        case SDL_MOUSEBUTTONUP:
            if(event_MouseRelease!=NULL)
                event_MouseRelease(this,form_ptr);
            break;
    }
```

滑鼠不再此 GameObj 內，應觸發所屬的 MouseLeave 事件，於是我就呼叫前面宣告的 function pointer 而參數就是 this,和 form 的 pointer

概念同，觸發滑鼠進入事件

概念同，觸發滑鼠按下事件

概念同，觸發滑鼠釋放事件

這樣做的好處就是可以更靈活運用控制項。同一種控制項，我只要把他相對應的 function pointer 指到我寫的 function，就可以輕鬆撰寫他各種事件所對應的行為。

在 frm_login.h 宣告針對按鈕的一先事件（滑鼠滑過，滑鼠離開，滑鼠按下）

```
static void btn_login_leave(void* ,void *);
static void btn_login_enter(void* ,void *);
static void btn_login_pressed(void* ,void *);
```

這些是按鈕實際要執行的程式

```
frm_Login::frm_Login(Game *s) :
    Form(s),txt_username(s,this),txt_
    ate(s,this),btn_login(s,this)

}
```

在 constructor initialize 時把
把 Game 和 frm_pointer 傳入

在 createObj()裡，

```
btn_login.setEventHandler_MouseEnter(btn_login_enter);
btn_login.setEventHandler_MouseLeave(btn_login_leave);
btn_login.setEventHandler_MousePressed(btn_login_pressed);
```

利用在 GameObj 裡宣告的 set function，把相對應的 function bind 到相對應的 function pointer，

```
static void btn_login_pressed(void* ,void *);
```

這裡有一點很重要：如果不用 static，則此 function 就是 class frm_Login 的 member function
因為我寫的 function pointer 是由 GameObj 裡的 handleEvent 來呼叫，但 function pointer 所指到的 function 是屬於 class frm_Login 的。而我們知道當 member function 被呼叫時還會自動傳入 this 作為參數。這樣程式編譯不會過，因為等於從 GameObj 的 handleEvent 呼叫另一個 class 的 member function。
所以我的解決方式就是用 static 宣告，但這又會碰到一個問題，就是 static function 不屬於 class 內，不能使用 class 的 private 成員。而當我按下登入，我當然會希望可以更改一些 frm_Login 的成員，還有自己本身(class GameObj)的成員。因此，我自己設指標，取代 this 指標的功能，把 frm_Login 和 GameObj 的 pointer 自己手動傳入，傳入後在 cast 成他應有的型態，如此一來就可以像是 member function 一樣存取 class member。

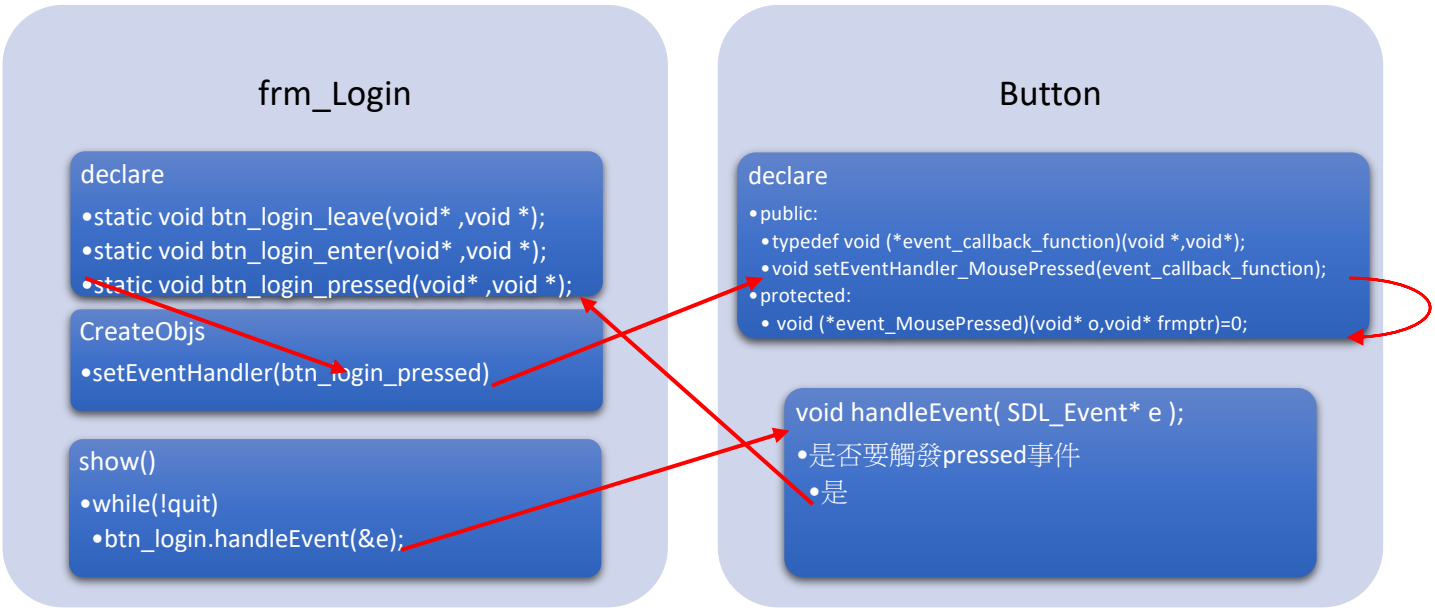
以下用登入按鈕按下時作為例子

```
void frm_Login::btn_login_pressed(void *b, void *s) {
    frm_Login *frmPtr = (frm_Login*)s;
    Button * obj= (Button*)b;
    std::string username,password,err;
    username = frmPtr->txt_username.getText();
    password = frmPtr->txt_password.getText();
    if(frmPtr->g->verifyAccount(username,passwo
    {
        frmPtr->g->login_status= true;
        //frmPtr->login_success = true;
        frmPtr->close();
    }else{
```

把 frm_Login 和 Button cast 回來

因為我把 frm_Login cast 回來，所以我可以存取同屬於 form 裡的其他控制項，就像是用 this 一樣

整個呼叫流程大約是這樣：



巧思 使用 double linked list 實現滾動選單：

Scroll List 部分，為了做出可滾動的選單，我設計了一個資料結構，經由上機作業(double linked list 修改)。我的想法是：在 Scroll List 中建立 speed 變數，當滑鼠滾輪向下滾時，speed 為一個正數和滾動速度有關。每次 render 時，先把每隔項目的 y 座標+=speed，作為位移。再來做判斷，如果最上方的項目足夠塞下一個項目，那我就將最後的項目一到最前面。往上滾亦然，只是換偵測下方空白。爲了實踐，我把所有的選單項目丟到自己建立的 double linked list。在裡面我加了幾個 function

(myqueue.h)

```
template<typename NODETYPE>
class List{
public:
    List();
    ~List();
    void insertAtFront(const NODETYPE &);
    void insertAtBack(const NODETYPE &);
    bool removeFromFront(NODETYPE &);
    bool removeFromBack(NODETYPE &);
    bool isEmpty() const;
    void offset(int);
    int getLength() const;
    NODETYPE * operator[] (int);
    NODETYPE * back();
    NODETYPE * front();
```

位移

取得第 n 項的指標

如果使用傳統陣列，要位移一項，全部的項目都要動，double linked list 只要動首尾附近

```cpp
template<typename T>
void Myqueue<T>::offset(int x) {
    if(x>0)
    {
        for(int i=0;i<x;++i)
        {
            T temp;
            this->removeFromBack(temp);
            this->insertAtFront(temp);
        }
    }else if(x<0)
    {
        for(int i=0;i<-x;++i)
        {
            T temp;
            this->removeFromFront(temp);
            this->insertAtBack(temp);
        }
    }
}
```

往下移

從後面移到前面

從前面移到後面

由於使用 double linked list 所以全部 offset 不用每一個 Node 都動到

operator overloading

```cpp
template<typename NODETYPE>
NODETYPE *List<NODETYPE>::operator[](int i) {
    ListNode<NODETYPE> *currentPtr = firstPtr;
    int u=0;
    while(currentPtr != 0)
    {
        if(u==i)
            break;
        currentPtr = currentPtr->nextPtr;
        ++u;
    }
    return currentPtr->getDataPtr();
}
```

由於是 double linked list
所以要找到第 n 項要用回圈
有了這個 overloading，取第 n 項就比較方便

```cpp
void operator <<(int);
void operator >>(int);
```

```cpp
template<typename NODETYPE>
void List<NODETYPE>::operator <<(int u){
    offset(-u);
}
template<typename NODETYPE>
void List<NODETYPE>::operator >>(int u){
    offset(u);
}
```

這邊也可以用 operator overloading，使得呼叫更方便
Ex:整個 List 向右移 1 項目：list>>1
PS 這段是後來加的沒有在最初的版本內

網路連線：

這部分的實作概念是由於之前用過 line API 所啟發。

建立一些 API endpoints 然後再使用 POST method，不過我沒有設計 access-token 之類的安全措施

本來是打算自己寫一個 tcp server side 的程式來處理 client 的需求，但最後考量到穩定性和時間的關係，我直接用 Apache，於是我用 port 80 來通訊。後端處理就用 php 寫簡單的網頁和 mysql 的連接，查詢。

我是使用我家電腦 ubuntu server 上的 apache，mysql 來儲存使用者資料。

每一項功能就寫成一個網頁，參數用 POST 傳入，我設計的功能有這些

create_new_user.php　　update_user_last_roles.php

reset_password.php　　update_user_roles.php

update_game_level.php　　verify.php

update_story_level.php

我的連線方式如下：(以登入說明)

我把使用 winsock 的程式碼放在 ServerRequest.h，並封裝成簡單的 class，以便呼叫。

```
class ServerRequest
{
    public:
        bool sendCmd(std::string &request_path,std::string &postdata,std::string &err_msg,std::string &filename);
    private:

};
```

傳送 http Request，並把 response 下載成檔案

利用 tcp socket 向我家的 server port 80 使用 POST method

以下為登入驗證的請求：↓

```
POST /game/verify.php HTTP/1.1
Host: www.atosystem.org
Content-type:application/x-www-form-urlencoded
Content-Length: 15
Connection: Close

n=username&p=password
```

我會把 server response 存成 account.txt，以供本地端使用

每次登入後，從 server 下載最新的資料存成檔案，遊戲進行中的時候，若有分數變動，先更新 local 檔案，等遊戲結束後在全部更新到 server。

以下為 Server response：↓

```
HTTP/1.1 200 OK
Date: Sun, 06 Jan 2019 08:47:57 GMT
Server: Apache/2.4.29 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 156
Connection: close
Content-Type: text/html; charset=UTF-8

UserID 1
username test
password aaa
level 6
roles 9 1 2 3 4 5 6 7 8 9
lastrole 1 2 3 4 5

game_progress
1 90 5
2 80 10
4 20 6
10 1 1
end_of_data
```

以下是以驗證帳號的 function 為例(節錄)：(Game.h)

```
bool Game::verifyAccount(std::string &u, std::string &p,std::string &errormsg) {

    std::string all =  "n=" + u + "&p=" + p;
    ServerRequest s;
    std::string s_path,s_cmd,s_file
    s_path = "/game/verify.php";
    s_cmd = all;
    s_filename = "account.txt";
    s.sendCmd(s_path,s_cmd,errormsg,s_filename);
    std::ifstream is("account.txt");
    std::string stuff;
    while(!is.eof()) {
        is>>stuff;
        if(stuff=="no_user_found") {
            errormsg = "no user found";
            return false;
        }
    }
```

把 POST Data 串在一起

指定目標網頁

指定下載 Response 檔案名稱

再將檔案讀入做後續認證處理

遊戲流程控制：

Main.cpp 主要撰寫一些視窗，Renderer 的初始化，控制畫面顯示

以下是顯示登入畫面的區塊

Main.cpp

```
{
    //login();
    frm_Login a(&g);
    a.show();
}
```

只要把 class Game 的 pointer 傳入(此處&g)，再用 show()，即可顯示在視窗中

之所以使用上下括號包住，是為了當 show()結束時，也就是視窗已關閉後，我利用這個方式自動呼叫 frm_Login 的 destructor，便於清空記憶體

主程式流程，我使用 nested while loop 寫，以下範例(此非專題內程式，只是說明)

```
while (!g.exit_game_flag) {
    int x;
    {
        Frm_somthing f(&g);
        f.show();
        x = f.getReturnCmd();
    }
    if(x==-2)
    {
        break;
    } else{
        {
            Frm_somthing f(&g);
            f.show();
            x = f.getReturnCmd();
        }
    }
}
```

每次檢查 exit_game_flag，如果再某個 form 中想要結束整個程式(Ex;視窗關閉被點擊)，就把 Game 中的 exit_game_flag 設為 true

宣告一個畫面物件，並叫他顯示，結束後再取一些參數(每個 Form 可自行定義)

若要回到上一個畫面只破壞目前前的迴圈即可

進入下一個面

心得與感想：

首先，我真的很感激教授當初同意我加簽這堂課。讓我有機會釐清一些之前不懂的觀念，也提供我實踐的機會。在這次的專題中，我試著把我之前寫其他程式的經驗用到 c++裡，並初步整合，Mysql，php，c++ tcp socket。我試著把 vb.net 的 Form 和 Form 上控制項的概念以自己所學來建立繼承關

係，並把 function bind 到指定的控制項的 event。這樣的設計思維的確為我帶來不少便利，但同時也因為把很多通用的參數(但後來實作之後發現有些 derived class 用不太到)丟到 base class，導致我只要建立少許的控制項，就佔了不少空間，這點我想是我還要再加強的地方。而寫 Callback 機制時，使我更了解 static function 和 member function 差異。當然，其中有許多地方事後想想，發現很多地方可以更精簡。

而我我想我們這組最大的問題就是沒有整合成功，只能在同學電腦上跑。雖然我們已經花很多時間找問題，但我們這組最後遊戲無法順利執行。我覺得在記憶體空間上的使用技巧可能是有待加強的地方。(常常卡在字體的部分，不明原因)

至於風格的部分，我是以另外一位同學所寫的劇情來設計，把城堡，酒吧，羊皮紙，酒瓶的元素融合在畫面裡。而音樂的部分則是使用鋼琴，中間再加上雙簧管最後是弦樂器，使用以小調為主在穿插一些大調部分增加張力，製造一點神秘，古老，蒼涼的氣氛，以搭配畫面背景。

總而言之，做完這次的專題，真的碰到許多問題，也學了不少東西，遺憾的就是無法解決程式合併後當掉的問題。此外，這份報告還是有蠻多地方可能寫得不是很好，不是很通順，請見諒。