**目錄**

[**一、** **簡介** - 1 -](#_Toc43829020)

[1. 動機 - 1 -](#_Toc43829021)

[2. 分工 - 1 -](#_Toc43829022)

[二、 **遊戲介紹** - 2 -](#_Toc43829023)

[1. 遊戲說明 - 2 -](#_Toc43829024)

[a. 遊戲規則 - 2 -](#_Toc43829025)

[b. 遊戲玩法 - 2 -](#_Toc43829026)

[2. 遊戲圖形 - 3 -](#_Toc43829027)

[a. 地圖 - 3 -](#_Toc43829028)

[b. 角色 - 3 -](#_Toc43829029)

[c. 敵人 - 3 -](#_Toc43829030)

[d. 遊戲封面 - 4 -](#_Toc43829031)

[e. 遊戲規則 - 4 -](#_Toc43829032)

[f. 計數器 - 5 -](#_Toc43829033)

[g. 血條 - 5 -](#_Toc43829034)

[3. 遊戲音效 - 7 -](#_Toc43829035)

[**三、** **程式設計** - 7 -](#_Toc43829036)

[1. 程式架構 - 7 -](#_Toc43829037)

[2. 程式類別 - 8 -](#_Toc43829038)

[四、 結語 - 10 -](#_Toc43829039)

[1. 問題及解決方法 - 10 -](#_Toc43829040)

[2. 時間表 - 11 -](#_Toc43829041)

[3. 貢獻比例 - 12 -](#_Toc43829042)

[4. 自我檢核表 - 12 -](#_Toc43829043)

[5. 收穫 - 13 -](#_Toc43829044)

[6. 心得 - 14 -](#_Toc43829045)

[**五、** **附錄** - 15 -](#_Toc43829046)

1. **簡介**
   1. 動機

會選擇這個遊戲來當作主題首先是因為覺得塔防遊戲很有趣，遊玩起來有些許難度，而且隨著塔或角色的種類和功能多樣化以後，可以延伸出更多的策略思路、隊伍搭配和技能組合，並且可能需要對應特定的關卡或敵人來採取不同的戰術。再來就是在考慮和權衡了遊戲製作的難度以後，認為這類型的遊戲在我們的能力範圍內，所以就決定以它來當作這堂課的目標。

* 1. 分工

顧軒文：程式設計開發、音效管理、檔案管理、報告撰寫

李振豪：美術素材蒐集，動畫素材製作、報告撰寫，遊戲大綱

1. **遊戲介紹**
   1. 遊戲說明
      1. 遊戲規則

這是一款單人的塔防類遊戲，玩家必須阻止敵人進入我方基地來獲勝。

* + 1. 遊戲玩法

進入遊戲關卡後，敵人便會從敵方基地沿著地圖上的路徑向我方基地移動，每當敵人進入我方基地，螢幕上方的記數便會減少，當記數減少至0時則玩家方落敗，而當地圖上所有敵人都被擊敗而記數尚未歸零時玩家方勝利。

要阻止敵方的目的就必須消耗顯示在螢幕右側的部署費用來部署角色，而每個角色的攻擊範圍可能都不同，而且面向也會影響攻擊範圍的覆蓋，先從右下角選擇角色，在欲放置角色的格子按住滑鼠左鍵，向欲面對的方向拖曳並放開完成部署。

地圖格子分為地面和高台，近戰角色只能部署在地面格子，而遠程角色則相反。地面角色的特殊機制-阻擋，能夠擋住附近敵人的移動，不過敵人也同時會攻擊阻擋自己的角色，若我方角色血量歸零則死亡，阻擋功能也會消失。

作弊功能：關卡中按下Space將直接使部署費用增加至99上限。

* 1. 遊戲圖形
     1. 地圖



分為地面區塊、高台區塊以及不能部署角色的高台區塊，用這3種格子來組成地圖的主體。



我方與敵方基地，明確表示了敵人行動的起點和終點，顯眼的顏色可以協助玩家制定防守策略。

* + 1. 角色



右下方選擇角色的圖標，由左到右用部署非用的大小來排序。



每個角色在遊戲地圖上對應的小人模型。

* + 1. 敵人



最基本的敵人種類，上圖為部分行走的動畫。



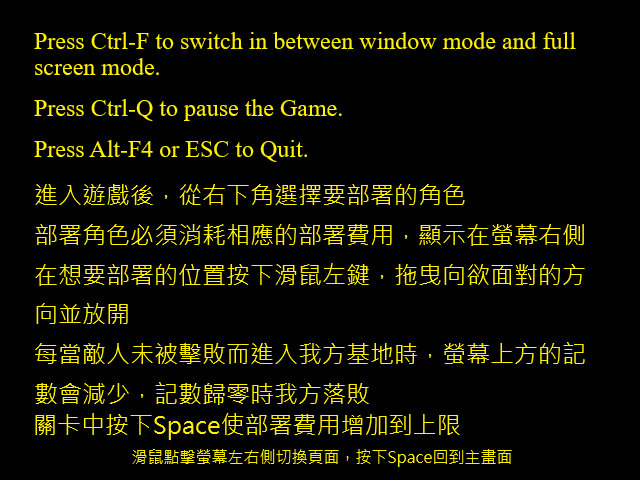
不同種類的敵人，有著移動速度極快的特性。

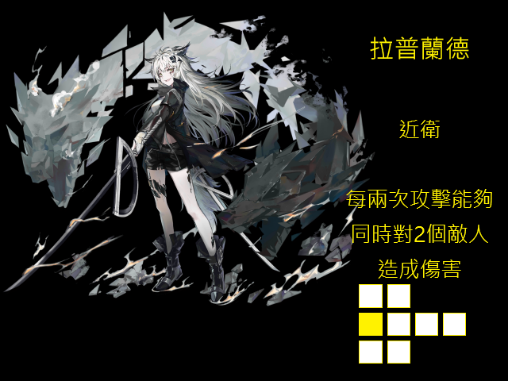
* + 1. 遊戲封面

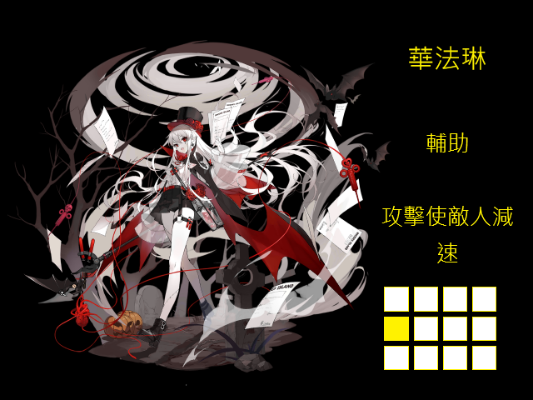


選擇這一張圖片來當作主畫面，富有空間感容易吸引人。

* + 1. 遊戲規則









用純黑的背景來凸顯角色立繪和黃色的文字，用較大的間隔來區分名字、職業、技能和攻擊範圍。

* + 1. 計數器



用來顯示部署費用和我方基地的生命值。

* + 1. 血條



敵人和我方角色都有血條的顯示，紅色表示失去的血量，綠色表示剩餘的血量。

* + 1. 飛行投射物





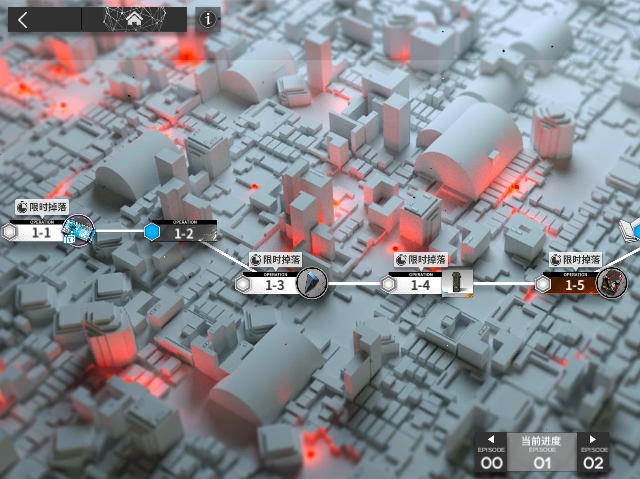
作為各個角色攻擊時發出的子彈來使用

* + 1. 關於我們



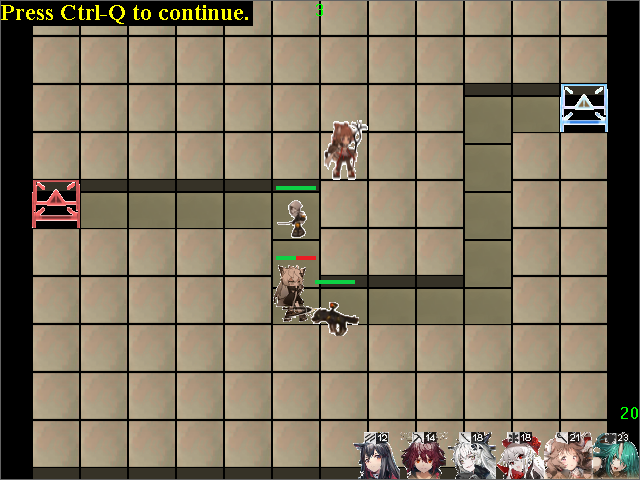
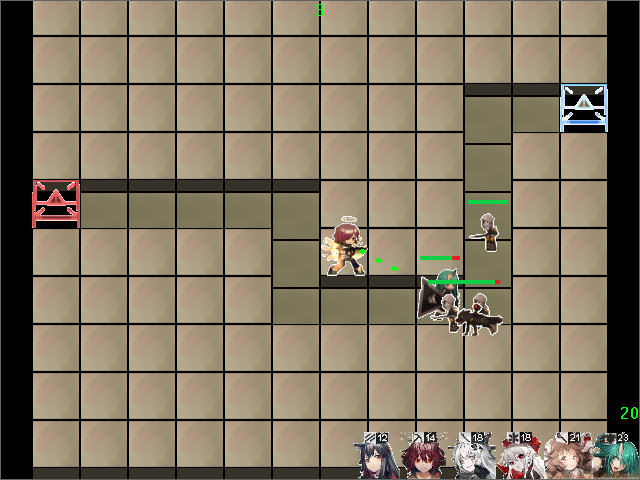
擷取課程網站上的一部分授課教師資訊，再加上所有組員的姓名、班級和聯絡方式。

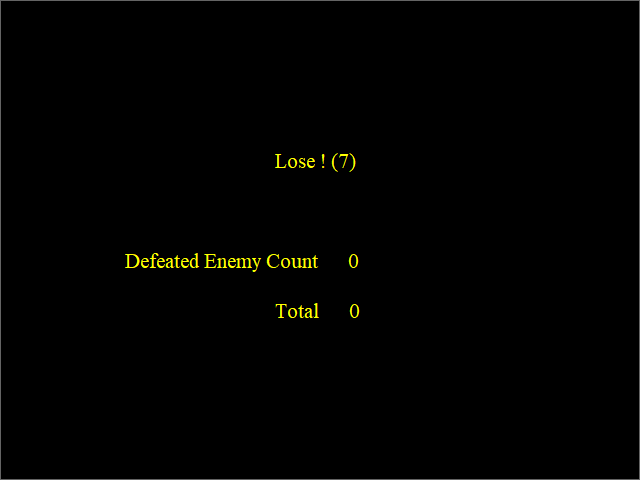
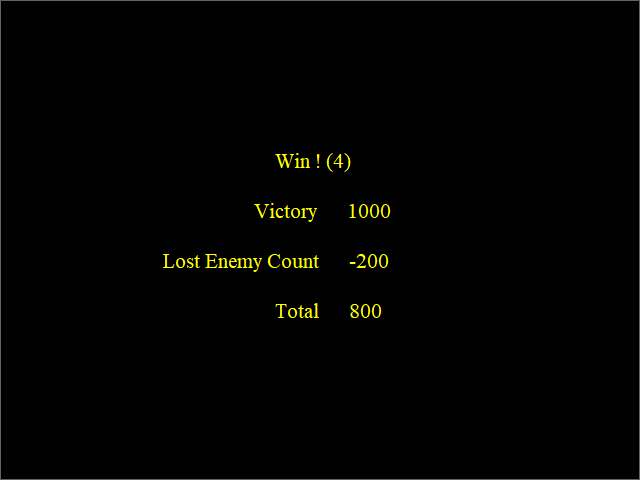
* + 1. 關卡選擇頁面



從主畫面選擇繼續後來到此畫面，可以點選關卡圖標的位置進入關卡。

* + 1. 遊戲畫面



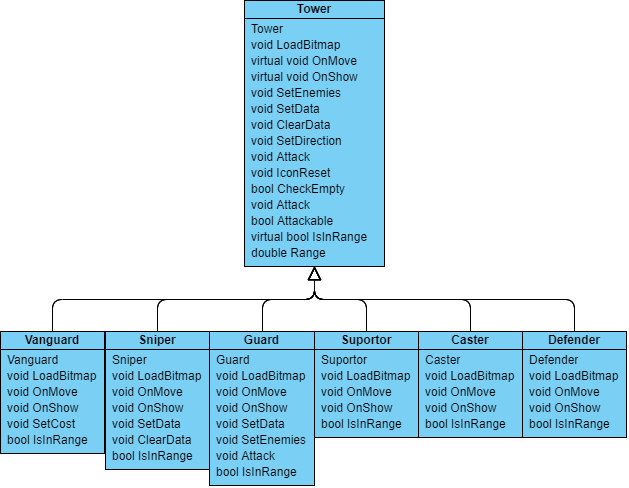


* 1. 遊戲音效

|  |  |  |
| --- | --- | --- |
| 音效名稱 | 音效使用時機 | 音效說明 |
| LevelChoosing | 關卡選擇頁面至關卡開始 | 略帶輕快的音樂。 |
| Level1 | 關卡1開始至結束 | 較為激昂的音樂，展現戰爭時期的緊張感 |

1. **程式設計**
   1. 程式架構

繼承部分



非繼承部分

|  |  |
| --- | --- |
| Class名稱 | 使用此Class的地方 |
| CGameMap | CGameStateRun |
| Tower | CGameStateRun |
| Projectile | 所有Tower類別 |
| Enemy | 所有Tower類別, CGameStateRun, LossCounter |
| Cost | CGameStateRun, Vanguard |
| LossCounter | CGameStateRun |
| Introduction | CGameStateInit |
| HpBar | Enemy, Vanguard, Guard, Defender |
| Result | CGame |
| CGame | CGameStateRun, CGameStateOver |

* 1. 程式類別

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| CGame | 3 | 9 | 引用Result |
| mygame | 102 | 491 | 控制遊戲主體 |
| Enemy | 24 | 109 | 控制敵人的行動和動畫顯示 |
| Tower | 31 | 117 | 所有角色的Base Class |
| Vanguard | 13 | 116 | 複寫和新增Base Class沒有的功能 |
| Sniper | 14 | 93 | 同上 |
| Guard | 18 | 155 | 同上 |
| Suportor | 10 | 80 | 同上 |
| Caster | 10 | 87 | 同上 |
| Defender | 10 | 109 | 同上 |
| HpBar | 11 | 41 | 控制血條正確顯示 |
| Projectile | 28 | 150 | 使子彈或投射物朝特定目標移動並對敵人造成傷害 |
| Cost | 14 | 54 | 控制部署費用的增減和顯示 |
| LossCounter | 17 | 69 | 控制我方基地生命值的計算 |
| Introduction | 13 | 38 | 控制Introduction頁面切換和顯示 |
| Total | 318 | 1718 |  |

1. 結語
   1. 問題及解決方法
      1. Class互相引用次數過多，編譯階段出現錯誤。
      2. 不知道如何在Over階段顯示Run階段的結果。
      3. OnLButtonDown中部署角色的部分重複的Code太多。
      4. 組員間互相配合的問題，如果有一方沒空的話，當周的程式進度或素材就會拖延。
      5. Projectile飛行路徑不正常

解決方法：

1. 釐清每個類別的作用，調整include的順序並在部分程式加上ifndef來防止重複宣告
2. 建立Result類別，從CGame中存取並調用修改就能同時在CGameStateRun和CGameStateOver中使用。
3. 優化程式碼，用一個Tower的指標指向被選擇的角色，便可以直接使用此指標而調用不同的角色，減少重複的程式碼
4. 將當周要處理的內容儘早交給對方，並事先確認時間上的配合。
5. 尚未完全解決，最剛開始子彈根本無法正確地往目的地移動，經過多次改動後，雖然能抵達目的地，但中途路徑可能會發生不正常的抖動。
   1. 時間表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 周次 | 顧軒文(小時) | 李振豪(小時) | 說明 | |
| 1 | 2 | 1 | 找素材，做地圖框架 | |
| 2 | 6 | 3 | 巡路邏輯，人物完整走完固定路線 | |
| 3 | 7 | 3 | 製作"塔"的class，對敵人進行攻擊 | |
| 4 | 5 | 3 | 做敵人的血量、塔的攻擊範圍 | |
| 5 | 5 | 5 | 可以自由選擇塔的位置 | |
| 6 | 5 | 3 | 基本UI，加上圖片素材 |
| 7 | 4 | 4 | 新增敵人血量顯示的UI，做第一個角色技能。 | |
| 8 | 4 | 4 | 優化鎖定系統，新增敵人動畫。 | |
| 9 | 5 | 3 | 新的角色(AOE傷害)和技能 | |
| 10 | 5 | 3 | 增加費用機制(消耗費用放置角色，第三個角色 | |
| 11 | 3 | 3 | 修改載入畫面，新增關卡選擇介面 | |
| 12 | 4 | 3 | 修改載入畫面，角色的面對方向，攻擊動畫，第四個角色 | |
| 13 | 3 | 3 | 角色的面對方向，攻擊動畫，設定勝敗條件 | |
| 14 | 6 | 3 | 攻擊動畫，第5個角色，新敵人 | |
| 15 | 12 | 7 | 第6個角色，新地圖，標題畫面，About訊息，操作說明 | |
| 合計 | 76 | 51 |  | |

* 1. 貢獻比例

顧軒文 60%

李振豪 40%

* 1. 自我檢核表

|  |  |  |  |
| --- | --- | --- | --- |
|  | 項目 | 完成否 | 無法完成的原因 |
| 1 | 解決Memory leak | 已完成 |  |
| 2 | 自訂遊戲 Icon | 已完成 |  |
| 3 | 全螢幕啟動 | 已完成 |  |
| 4 | 有About畫面 | 已完成 |  |
| 5 | 初始畫面說明按鍵及滑鼠之用法與密技 | 已完成 |  |
| 6 | 上傳 setup/apk/source檔 | 已完成 |  |
| 7 | setup檔可正確執行 | 已完成 |  |
| 8 | 報告字型、點數、對齊、行距、頁碼、等格式正確 | 已完成 |  |
| 9 | 報告封面、側邊格式正確 | 已完成 |  |
| 10 | 報告附錄程式格式正確 | 已完成 |  |

* 1. 收穫

顧軒文：這整個學期中本堂課讓我學習到不只是遊戲，而是一整個應用程式的結構和組成方式，也在真正意義上了解了物件導向這4個字，若是不將程式做成物件的形式，整個專案會顯得非常雜亂。我還意識到有很多東西和表面上看到的不一樣，一個簡單的功能也許藏著大量的細節要去注意，而看似複雜的功能在設計層面卻可能意外地容易製作。這次比較大規模的實際運用了繼承的方法讓我發現繼承不但能方便工程師去撰寫，有效減少重複程式碼，更能讓並非開發者的人一眼就能看出Class的用途和使用方式。

* 1. 心得

顧軒文：我覺得從頭設計一個完整的應用程式很不簡單，如果沒有老師提供的框架，我們在這學期一定不能有現在的進度，不過事實上，我們的遊戲也不算是完全體，後續還能夠再繼續擴充，例如更多的可用角色、新種類的敵人，甚至是Boss，更有趣的關卡等等，也還有不少潛在的問題需要解決，但我藉此機會學到了很多東西，不管是在程式設計或是其他方面，作為我們設計原型的這款塔防遊戲也是我十分喜歡的一款，未來有機會也許會繼續增加這個專案的完成度。

1. **附錄**

gamelib.h

class Result {

public:

R result;

int totalEnemyCount;

int lossEnemyCount;

};

class CGame {

private:

Result result;

};

gamelib.cpp

void CGame::SetResult(int result, int totalCount, int lossCount) {

this->result.result = (R)result;

this->result.totalEnemyCount = totalCount;

this->result.lossEnemyCount = lossCount;

}

Result CGame::GetResult() {

return result;

}

mygame.h

#include "Enemy.h"

#include "Projectile.h"

#include "Tower.h"

#include "Vanguard.h"

#include "Sniper.h"

#include "Guard.h"

#include "Caster.h"

#include "Suportor.h"

#include "Defender.h"

#include "Cost.h"

#include "LossCounter.h"

#include "Introduction.h"

namespace game\_framework {

enum AUDIO\_ID { // 定義各種音效的編號

AUDIO\_LEVELCHOOSING,

AUDIO\_LEVEL1,

AUDIO\_LEVEL2

};

class CGameMap {

public:

CGameMap();

void LoadBitmap();

void OnShow();

void SetLevel(int);

int map1[10][12];

int map2[10][12];

protected:

CMovingBitmap start, ground, end, highground1, highground2;

const int x, y;

const int w, h;

int level;

};

class CGameStateInit : public CGameState {

public:

CGameStateInit(CGame \*g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

void OnKeyUp(UINT, UINT, UINT); // 處理鍵盤Up的動作

void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CMovingBitmap logo; // csie的logo

CMovingBitmap mainMenu, introButton, aboutButton, about;

Introduction intro;

enum State { Main, Intro, About }; State state;

};

class CGameStateRun : public CGameState {

public:

CGameStateRun(CGame \*g);

~CGameStateRun();

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit(); // 遊戲的初值及圖形設定

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnLButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnMouseMove(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnRButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnRButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

const int NUMBALLS; // 球的總數

CMovingBitmap background; // 背景圖

CMovingBitmap help; // 說明圖

CMovingBitmap levels;

CAnimation loading;

CGameMap gamemap;

vector<Enemy> enemies{ Enemy(0),Enemy(1), Enemy(0), Enemy(1), Enemy(0) };

Vanguard Texas;

Sniper Exusiai;

Guard Lappland;

Caster Eyjafjalla;

Suportor Warfarin;

Defender Hoshikuma;

Tower \*towerSelected = nullptr; //用來指向其中一個角色，優化程式碼

Cost cost;

LossCounter lossCounter;

CPoint clickPoint;

enum State {Level, Game }; State state;

enum Levels { None, level1, level2 }; Levels level;

};

class CGameStateOver : public CGameState {

public:

CGameStateOver(CGame \*g);

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit();

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

int counter; // 倒數之計數器

};

}

mygame.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

#include "string"

#include <time.h>

namespace game\_framework {

CGameMap::CGameMap() :x(32), y(0), w(48), h(48) {

int map1\_init[10][12] = { {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},

{4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},

{4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 2, 1},

{4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 4},

{3, 2, 2, 2, 2, 2, 4, 4, 4, 2, 4, 4},

{4, 4, 4, 4, 4, 2, 4, 4, 4, 2, 4, 4},

{4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 4, 4},

{4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},

{4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},

{4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4} };

for (int i = 0; i < 10; i++)

for (int j = 0; j < 12; j++)

map1[i][j] = map1\_init[i][j];

}

void CGameMap::LoadBitmap() {

start.LoadBitmap(IDB\_START, RGB(255, 255, 255));

ground.LoadBitmap(IDB\_GROUND);

end.LoadBitmap(IDB\_END, RGB(255, 255, 255));

highground1.LoadBitmap(IDB\_HIGHGROUND1);

highground2.LoadBitmap(IDB\_HIGHGROUND2);

}

void CGameMap::OnShow() {

CMovingBitmap \*bitmap;

int correction;

for (int i = 0; i < 10; i++)

for (int j = 0; j < 12; j++) {

correction = 0;

switch (map1[i][j])

{

case 0:

break;

case 1:

bitmap = &start;

correction = -12;

break;

case 2:

bitmap = &ground;

break;

case 3:

bitmap = &end;

correction = -12;

break;

case 4:

bitmap = &highground1;

correction = -12;

break;

case 5:

bitmap = &highground2;

correction = -12;

break;

default:

ASSERT(0);

}

bitmap->SetTopLeft(x + (w\*j), y + (h\*i) + correction);

bitmap->ShowBitmap();

}

}

void CGameMap::SetLevel(int level) {

this->level = level;

}

CGameStateInit::CGameStateInit(CGame \*g)

: CGameState(g)

{

}

void CGameStateInit::OnInit()

{

ShowInitProgress(0); // 一開始的loading進度為0%

Sleep(300);

mainMenu.LoadBitmap(IDB\_MAINMENU);

intro.LoadBitmap();

about.LoadBitmap(IDB\_ABOUT);

introButton.LoadBitmap(IDB\_INTRO\_BUTTON, RGB(255, 255, 255));

aboutButton.LoadBitmap(IDB\_ABOUT\_BUTTON, RGB(255, 255, 255));

}

void CGameStateInit::OnBeginState()

{

state = Main;

}

void CGameStateInit::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)

{

const char KEY\_ESC = 27;

const char KEY\_SPACE = ' ';

switch (state)

{

case Main:

if (nChar == KEY\_SPACE)

GotoGameState(GAME\_STATE\_RUN);

else if (nChar == KEY\_ESC)

PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_CLOSE, 0, 0); // 關閉遊戲

break;

case Intro:

if (nChar == KEY\_SPACE)

state = Main;

break;

case About:

if (nChar == KEY\_SPACE)

state = Main;

break;

}

}

void CGameStateInit::OnLButtonDown(UINT nFlags, CPoint point)

{

switch (state)

{

case Main: //判斷是否按下introduction或about

if (point.x >= 480 && point.x <= 640 && point.y >= 0 && point.y <= 50)

state = Intro;

else if (point.x >= 480 && point.x <= 640 && point.y >= 50 && point.y <= 100)

state = About;

else

GotoGameState(GAME\_STATE\_RUN); // 切換至GAME\_STATE\_RUN

break;

case Intro:

if (point.x <= 320) //翻頁

intro.IndexMinus();

else

intro.IndexAdd();

break;

case About:

break;

}

}

void CGameStateInit::OnShow()

{

mainMenu.SetTopLeft(0, 0);

mainMenu.ShowBitmap();

introButton.SetTopLeft(480, 0);

introButton.ShowBitmap();

aboutButton.SetTopLeft(480, 50);

aboutButton.ShowBitmap();

CDC \*pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \*fp;

f.CreatePointFont(160, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0, 0, 0));

pDC->SetTextColor(RGB(255, 255, 0));

pDC->TextOut(120, 425, "Please click mouse or press SPACE to begin.");

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

switch (state)

{

case Main:

break;

case Intro:

intro.OnShow();

break;

case About:

about.ShowBitmap();

break;

}

}

CGameStateOver::CGameStateOver(CGame \*g)

: CGameState(g)

{

}

void CGameStateOver::OnMove()

{

counter--;

if (counter < 0)

GotoGameState(GAME\_STATE\_INIT);

}

void CGameStateOver::OnBeginState()

{

counter = 30 \* 10;

CAudio::Instance()->Stop(AUDIO\_LEVEL1);

CAudio::Instance()->Stop(AUDIO\_LEVEL2);

}

void CGameStateOver::OnInit()

{

ShowInitProgress(66); // 接個前一個狀態的進度，此處進度視為66%

Sleep(300); // 放慢，以便看清楚進度，實際遊戲請刪除此Sleep

ShowInitProgress(100);

}

void CGameStateOver::OnShow()

{

CDC \*pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \*fp;

f.CreatePointFont(160, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0, 0, 0));

pDC->SetTextColor(RGB(255, 255, 0));

char result[80], str[80],str2[80];

int totalEnemyCount = game->GetResult().totalEnemyCount,

lossEnemyCount = game->GetResult().lossEnemyCount;

if (game->GetResult().result == Lose) { //顯示計分

sprintf(result, "Lose ! (%d)", counter / 30);

sprintf(str, "Defeated Enemy Count %d", (totalEnemyCount - lossEnemyCount) \* 100);

sprintf(str2, "Total %d", (totalEnemyCount - lossEnemyCount) \* 100);

pDC->TextOut(275, 150, result);

pDC->TextOut(125, 250, str);

pDC->TextOut(275, 300, str2);

}

else if (game->GetResult().result == Win) {

sprintf(result, "Win ! (%d)", counter / 30);

pDC->TextOut(254, 200, "Victory 1000");

sprintf(str, "Lost Enemy Count %d", lossEnemyCount \* -100);

sprintf(str2, "Total %d", 1000 - (lossEnemyCount \* 100));

pDC->TextOut(275, 150, result);

pDC->TextOut(163, 250, str);

pDC->TextOut(275, 300, str2);

}

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

CGameStateRun::CGameStateRun(CGame \*g)

: CGameState(g), NUMBALLS(28)

{

}

CGameStateRun::~CGameStateRun()

{

}

void CGameStateRun::OnBeginState()

{

CAudio::Instance()->Play(AUDIO\_LEVELCHOOSING, true);

state = Level; //重置關卡

cost.Reset();

Texas.ClearData();

Exusiai.ClearData();

Lappland.ClearData();

Warfarin.ClearData();

Eyjafjalla.ClearData();

Hoshikuma.ClearData();

for (int i = 0; i < 5; i++) {

enemies[i].Reset();

}

}

void CGameStateRun::OnMove() // 移動遊戲元素

{

if (state == Game) {

cost.OnMove();

lossCounter.CheckEnemies(enemies);

if (lossCounter.CheckLost()) { //失敗後的結算

game->SetResult(Lose, enemies.size(), lossCounter.GetLossCount(enemies));

GotoGameState(GAME\_STATE\_OVER);

}

else if (lossCounter.CheckWin(enemies)) { //勝利的結算

game->SetResult(Win, enemies.size(), lossCounter.GetLossCount(enemies));

GotoGameState(GAME\_STATE\_OVER);

}

static int n = 0;

static clock\_t t = clock();

if (clock() - t >= 1000 && n != 4) {

t = clock();

n++;

}

for (int i = 0; i <= n; i++)

enemies[i].OnMove();

Texas.SetEnemies(enemies);

Texas.OnMove();

Exusiai.SetEnemies(enemies);

Exusiai.OnMove();

Lappland.SetEnemies(enemies);

Lappland.OnMove();

Warfarin.SetEnemies(enemies);

Warfarin.OnMove();

Eyjafjalla.SetEnemies(enemies);

Eyjafjalla.OnMove();

Hoshikuma.SetEnemies(enemies);

Hoshikuma.OnMove();

}

}

void CGameStateRun::OnInit() // 遊戲的初值及圖形設定

{

ShowInitProgress(33); // 接個前一個狀態的進度，此處進度視為33%

levels.LoadBitmap(IDB\_LEVELS);

gamemap.LoadBitmap();

cost = Cost(50);

cost.LoadBitmap();

lossCounter.LoadBitmap();

ShowInitProgress(50);

Sleep(300); // 放慢，以便看清楚進度，實際遊戲請刪除此Sleep

CAudio::Instance()->Load(AUDIO\_LEVELCHOOSING, "sounds\\LevelChoosing.mp3");

CAudio::Instance()->Load(AUDIO\_LEVEL1, "sounds\\Level1.mp3");

CAudio::Instance()->Load(AUDIO\_LEVEL2, "sounds\\Level2.mp3");

Texas.LoadBitmap();

Texas.SetCost(&cost.cost);

Exusiai.LoadBitmap();

Lappland.LoadBitmap();

Warfarin.LoadBitmap();

Eyjafjalla.LoadBitmap();

Hoshikuma.LoadBitmap();

for (int i = 0; i < 5; i++) {

enemies[i].LoadBitmap();

}

}

void CGameStateRun::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

}

void CGameStateRun::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)

{

const char KEY\_SPACE = ' ';

if (nChar == KEY\_SPACE) //開啟作弊模式

cost.CheatingMode();

}

void CGameStateRun::OnLButtonDown(UINT nFlags, CPoint point) // 處理滑鼠的動作

{

switch (state) {

case Level: //選擇關卡

if (point.x >= 10 && point.x <= 90 && point.y >= 220 && point.y <= 240) {

level = level1;

state = Game;

CAudio::Instance()->Stop(AUDIO\_LEVELCHOOSING);

CAudio::Instance()->Play(AUDIO\_LEVEL1);

gamemap.SetLevel(1);

}

else if (point.x >= 140 && point.x <= 220 && point.y >= 220 && point.y <= 240) {

level = level2;

state = Game;

CAudio::Instance()->Stop(AUDIO\_LEVELCHOOSING);

CAudio::Instance()->Play(AUDIO\_LEVEL1);

}

else if (point.x >= 8 && point.x <= 186 && point.y >= 7 && point.y <= 31) {

GotoGameState(GAME\_STATE\_INIT);

CAudio::Instance()->Stop(AUDIO\_LEVELCHOOSING);

CAudio::Instance()->Play(AUDIO\_LEVEL2);

}

break;

case Game: //部署角色

clickPoint = point;

Texas.IconReset();

Exusiai.IconReset();

Lappland.IconReset();

Warfarin.IconReset();

Eyjafjalla.IconReset();

Hoshikuma.IconReset();

static int index;

if (point.x >= 352 && point.y >= 432) {

index = (point.x - 352) / 48;

switch (index) {

case 5:

Hoshikuma.icon.SetTopLeft(592, 422);

break;

case 4:

Eyjafjalla.icon.SetTopLeft(544, 422);

break;

case 3:

Warfarin.icon.SetTopLeft(496, 422);

break;

case 2:

Lappland.icon.SetTopLeft(448, 422);

break;

case 1:

Exusiai.icon.SetTopLeft(400, 422);

break;

case 0:

Texas.icon.SetTopLeft(352, 422);

break;

}

OutputDebugString(\_T((to\_string(index) + " ").c\_str()));

}

else {

int temp\_x = (point.x - 32) / 48, temp\_y = (point.y + 12) / 48;

int x = (48 \* (temp\_x)) + 32, y = (48 \* (temp\_y)) - 24;

switch (index) {

case 5:

if (gamemap.map1[temp\_y][temp\_x] == 2 && cost.cost >= Hoshikuma.cost && Hoshikuma.CheckEmpty(x, y)) {

Hoshikuma.SetData(x, y, clock(), &cost.cost);

towerSelected = (Tower\*)(&Hoshikuma);

}

break;

case 4:

if (gamemap.map1[temp\_y][temp\_x] == 4 && cost.cost >= Eyjafjalla.cost && Eyjafjalla.CheckEmpty(x, y)) {

Eyjafjalla.SetData(x, y, clock(), &cost.cost);

towerSelected = (Tower\*)(&Eyjafjalla);

}

break;

case 3:

if (gamemap.map1[temp\_y][temp\_x] == 4 && cost.cost >= Warfarin.cost && Warfarin.CheckEmpty(x, y)) {

Warfarin.SetData(x, y, clock(), &cost.cost);

towerSelected = (Tower\*)(&Warfarin);

}

break;

case 2:

if (gamemap.map1[temp\_y][temp\_x] == 2 && cost.cost >= Lappland.cost && Lappland.CheckEmpty(x, y)) {

Lappland.SetData(x, y, clock(), &cost.cost);

towerSelected = (Tower\*)(&Lappland);

}

break;

case 1:

if (gamemap.map1[temp\_y][temp\_x] == 4 && cost.cost >= Exusiai.cost && Exusiai.CheckEmpty(x, y)) {

Exusiai.SetData(x, y, clock(), &cost.cost);

towerSelected = (Tower\*)(&Exusiai);

}

break;

case 0:

if (gamemap.map1[temp\_y][temp\_x] == 2 && cost.cost >= Texas.cost && Texas.CheckEmpty(x, y)) {

Texas.SetData(x, y, clock(), &cost.cost);

towerSelected = (Tower\*)(&Texas);

}

break;

}

}

break;

}

}

void CGameStateRun::OnLButtonUp(UINT nFlags, CPoint point) // 處理滑鼠的動作

{

if (towerSelected) {

towerSelected->IsActive = true;

int xOffset = point.x - clickPoint.x, yOffset = point.y - clickPoint.y;

if (abs(xOffset) > abs(yOffset)) { //面對方向

if (xOffset >= 0)

towerSelected->SetDirection(3);

else

towerSelected->SetDirection(2);

}

else {

if (yOffset >= 0)

towerSelected->SetDirection(1);

else

towerSelected->SetDirection(0);

}

towerSelected = nullptr;

}

}

void CGameStateRun::OnMouseMove(UINT nFlags, CPoint point) // 處理滑鼠的動作

{

}

void CGameStateRun::OnRButtonDown(UINT nFlags, CPoint point) // 處理滑鼠的動作

{

}

void CGameStateRun::OnRButtonUp(UINT nFlags, CPoint point) // 處理滑鼠的動作

{

}

void CGameStateRun::OnShow()

{

switch (state)

{

case Level:

levels.SetTopLeft(0, 0);

levels.ShowBitmap();

break;

case Game:

gamemap.OnShow();

Texas.OnShow();

Exusiai.OnShow();

Lappland.OnShow();

Warfarin.OnShow();

Eyjafjalla.OnShow();

Hoshikuma.OnShow();

for (int i = 0; i < 5; i++) {

enemies[i].OnShow();

}

cost.OnShow();

lossCounter.OnShow();

break;

default:

break;

}

}

}

Enemy.h

#ifndef HPBAR\_H

#include "HpBar.h"

#endif

namespace game\_framework {

class Enemy {

public:

Enemy();

Enemy(int);

Enemy(int, int);

void LoadBitmap();

void OnMove();

void OnShow();

bool isAlive(); //判斷此敵人是否還活著

void Reset(); //重新開始遊戲時要重置

CAnimation pic;

vector<pair<int, int>> path1{ make\_pair(12,3),make\_pair(10,3),make\_pair(10,7),make\_pair(6,7),make\_pair(6,5),make\_pair(1,5) };

int x, y, hp, index, speed, initSpeed, level;

enum PicType { minion, dog }; PicType picType;

bool isBlocked;

HpBar hpBar;

};

}

Enemy.cpp

#include "stdafx.h"

#include "resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include <string>

#include "Enemy.h"

namespace game\_framework {

Enemy::Enemy() {

picType = minion;

x = 32 + 48 \* (path1[0].first - 1);

y = 48 \* (path1[0].second - 1);

hp = 100;

index = 0;

isBlocked = false;

}

Enemy::Enemy(int type) {

picType = PicType(type);

x = 32 + 48 \* (path1[0].first - 1);

y = 48 \* (path1[0].second - 1);

hp = 100;

index = 0;

isBlocked = false;

}

Enemy::Enemy(int type, int level) {

picType = PicType(type);

x = 32 + 48 \* (path1[0].first - 1);

y = 48 \* (path1[0].second - 1);

hp = 100;

index = 0;

isBlocked = false;

this->level = level;

}

void Enemy::LoadBitmap() {

switch (picType)

{

case minion: {

char \*bmp[12] = { ".\\bitmaps\\man1.bmp", ".\\bitmaps\\man2.bmp", ".\\bitmaps\\man3.bmp", ".\\bitmaps\\man4.bmp", ".\\bitmaps\\man5.bmp", ".\\bitmaps\\man6.bmp", ".\\bitmaps\\man7.bmp", ".\\bitmaps\\man8.bmp", ".\\bitmaps\\man9.bmp", ".\\bitmaps\\man10.bmp", ".\\bitmaps\\man11.bmp", ".\\bitmaps\\man12.bmp" };

for (int i = 0; i < 12; i++)

pic.AddBitmap(bmp[i], RGB(255, 255, 255));

initSpeed = 2;

speed = 2;

}

break;

case dog: {

char \*bmp[6] = { ".\\bitmaps\\wolf1.bmp", ".\\bitmaps\\wolf2.bmp", ".\\bitmaps\\wolf3.bmp", ".\\bitmaps\\wolf4.bmp", ".\\bitmaps\\wolf5.bmp", ".\\bitmaps\\wolf6.bmp" };

for (int i = 0; i < 6; i++)

pic.AddBitmap(bmp[i], RGB(255, 255, 255));

initSpeed = 3;

speed = 3;

break;

}

}

hpBar.LoadBitmap();

}

void Enemy::OnMove() {

if (!isBlocked && isAlive()) {

if (x == 32 + 48 \* (path1[index + 1].first - 1) && y == 48 \* (path1[index + 1].second - 1) && index != 5) {

x = 32 + 48 \* (path1[index + 1].first - 1);

y = 48 \* (path1[index + 1].second - 1);

index++;

}

if (index != 5) {

if (y != 48 \* (path1[index + 1].second - 1)) {

if (y > 48 \* (path1[index + 1].second - 1))

y -= speed;

else

y += speed;

}

else if (x != 32 + 48 \* (path1[index + 1].first - 1)) {

if (x > 48 \* (path1[index + 1].first - 1))

x -= speed;

else

x += speed;

}

pic.OnMove();

}

}

pic.SetTopLeft(x, y);

hpBar.Set(x, y, hp, 100);

}

void Enemy::OnShow() {

if (isAlive() && index!=path1.size()-1) {

pic.OnShow();

hpBar.OnShow();

}

}

bool Enemy::isAlive() {

if (hp > 0)

return true;

return false;

}

void Enemy::Reset() {

x = 32 + 48 \* (path1[0].first - 1);

y = 48 \* (path1[0].second - 1);

hp = 100;

index = 0;

isBlocked = false;

}

}

Tower.h

namespace game\_framework {

class Tower {

public:

Tower();

void LoadBitmap();

virtual void OnMove();

virtual void OnShow();

void SetEnemies(vector<Enemy>&); //設定攻擊目標

void SetData(int, int, clock\_t, int\*); //設定新部署角色的資料

void ClearData(); //重新開始時清除資料

void SetDirection(int); //設定方向

void Attack(int); //攻擊

void IconReset(); //右下圖標重置位置

bool CheckEmpty(int, int); //檢查格子是否已有角色

bool Attackable(int,int); //是否可以進行攻擊

virtual bool IsInRange(int ,int ,int ,int) = 0; //是否在攻擊範圍內

double Range(int, int, int, int); //計算距離

CMovingBitmap icon, char\_l, char\_r;

Projectile projectile;

vector<pair<int, int>> pos;

vector<int> sp, blockCount, hp;

vector<clock\_t> time;

vector<Enemy\*> target, enemies, blockedEnemies;

bool findTarget, IsActive, IsSkillActive;

int MaxSp, iconX, cost, MaxBlockCount,MaxHp;

double atk\_speed;

enum Direction { Up, Down, Left, Right };

Direction direction;

HpBar hpBar;

};

}

Tower.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Enemy.h"

#include "Projectile.h"

#include "Tower.h"

#include <time.h>

#include <string>

namespace game\_framework {

Tower::Tower() {

findTarget = false;

IsActive = false;

IsSkillActive = false;

MaxHp = 100;

}

void Tower::LoadBitmap() {

char\_r.LoadBitmap(IDB\_EXUISI\_R, RGB(255, 255, 255));

projectile.LoadBitmap();

}

void Tower::SetEnemies(vector<Enemy>& enemies) {

for (int i = 0; i < pos.size(); i++) {

for (int j = 0; j < enemies.size(); j++) {

if (enemies[j].isAlive() && IsInRange(pos[i].first, pos[i].second, enemies[j].x, enemies[j].y)) {

this->target[i] = &(enemies[j]);

projectile.SetTarget(enemies[j]);

break;

}

}

}

if (this->enemies.empty())

for (int i = 0; i < enemies.size(); i++)

this->enemies.push\_back(&enemies[i]);

}

void Tower::SetData(int x, int y, clock\_t t,int \*cost) {

pos.push\_back(make\_pair(x, y));

time.push\_back(t);

sp.push\_back(0);

target.push\_back(0);

blockCount.push\_back(0);

hp.push\_back(MaxHp);

\*cost -= this->cost;

}

void Tower::ClearData() {

pos.clear();

time.clear();

sp.clear();

target.clear();

blockCount.clear();

hp.clear();

projectile.pos.clear();

}

void Tower::SetDirection(int dir) {

switch (dir)

{

case 0:

direction = Up;

break;

case 1:

direction = Down;

break;

case 2:

direction = Left;

break;

case 3:

direction = Right;

break;

}

}

bool Tower::CheckEmpty(int x, int y) {

for (int i = 0; i < pos.size(); i++)

if (pos[i].first == x && pos[i].second == y)

return false;

return true;

}

double Tower::Range(int x1, int x2, int y1, int y2) {

return sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2));

}

bool Tower::Attackable(int i,int t) {

return (clock() - time[i] >= t) ? true : false;

}

void Tower::Attack(int i){

projectile.SetPosition(pos[i].first + 24, pos[i].second + 24);

time[i] = clock();

}

void Tower::IconReset() {

icon.SetTopLeft(iconX, 432);

}

}

Vanguard.h

namespace game\_framework {

class Vanguard : public Tower {

public:

Vanguard();

void LoadBitmap();

void OnMove();

void OnShow();

void SetCost(int\*);

bool IsInRange(int, int, int, int);

private:

int \*currentCost;

};

}

Vanguard.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Enemy.h"

#include "Projectile.h"

#include "Tower.h"

#include "Vanguard.h"

#include <string>

namespace game\_framework {

Vanguard::Vanguard() : Tower() {

atk\_speed = 1.5;

MaxSp = 10;

iconX = 352;

cost = 12;

projectile.SetPicNumber(IDB\_PROJECTILE);

projectile.SetDmg(10);

MaxBlockCount = 2;

MaxHp = 150;

}

void Vanguard::LoadBitmap() {

icon.LoadBitmap(IDB\_TEXAS, RGB(255, 255, 255));

char\_l.LoadBitmap(IDB\_TEXAS\_L, RGB(255, 255, 255));

char\_r.LoadBitmap(IDB\_TEXAS\_R, RGB(255, 255, 255));

projectile.LoadBitmap();

hpBar.LoadBitmap();

Tower::IconReset();

}

void Vanguard::OnMove() {

if (IsActive) {

for (int i = 0; i < time.size(); i++) {

if (hp[i] > 0) {

if (target[i] != nullptr && Range(pos[i].first, target[i]->x, pos[i].second, target[i]->y) <= 48 \* 4 && target[i]->isAlive()) {

if (Attackable(i, 1000 / atk\_speed)) {

Attack(i);

sp[i]++;

hp[i] -= 10 \* blockCount[i];

}

if (sp[i] == MaxSp) {

\*currentCost += 10;

sp[i] = 0;

}

}

for (int j = 0; j < blockedEnemies.size(); j++) {

if (!blockedEnemies[j]->isAlive()) {

enemies[j]->isBlocked = false;

blockedEnemies.erase(blockedEnemies.begin() + j);

j--;

blockCount[i]--;

}

}

for (int j = 0; j < enemies.size(); j++) {

if (blockCount[i] < MaxBlockCount && enemies[j]->isAlive() && !enemies[j]->isBlocked && Range(pos[i].first, enemies[j]->x, pos[i].second, enemies[j]->y) < 48) {

enemies[j]->isBlocked = true;

blockedEnemies.push\_back(enemies[j]);

blockCount[i]++;

}

}

}

else {

for (int j = 0; j < blockedEnemies.size(); j++)

blockedEnemies[j]->isBlocked = false;

blockedEnemies.clear();

for (int j = 0; j < blockCount.size(); j++)

blockCount[j] = 0;

}

}

projectile.OnMove();

}

}

void Vanguard::OnShow() {

icon.ShowBitmap();

for (int i = 0; i < pos.size(); i++) {

if (hp[i] > 0) {

hpBar.Set(pos[i].first, pos[i].second, hp[i], MaxHp);

hpBar.OnShow();

}

}

Tower::OnShow();

}

void Vanguard::SetCost(int \*cost) {

currentCost = cost;

}

bool Vanguard::IsInRange(int selfX, int selfY, int targetX, int targetY) {

switch (int(direction))

{

case 0:

if (targetX >= selfX && targetX <= selfX + 48 && targetY >= selfY - 36 && targetY <= selfY + 60)

return true;

break;

case 1:

if (targetX >= selfX && targetX <= selfX + 48 && targetY >= selfY + 12 && targetY <= selfY + 108)

return true;

break;

case 2:

if (targetX >= selfX - 48 && targetX <= selfX + 48 && targetY >= selfY + 12 && targetY <= selfY + 60)

return true;

break;

case 3:

if (targetX >= selfX && targetX <= selfX + 96 && targetY >= selfY + 12 && targetY <= selfY + 60)

return true;

break;

}

return false;

}

}

Sniper.h

namespace game\_framework {

class Sniper : public Tower {

public:

Sniper();

void LoadBitmap();

void OnMove();

void OnShow();

void SetData(int, int, clock\_t, int\*);

void ClearData();

bool IsInRange(int, int, int, int);

private:

vector<int> skillHit;

};

}

Sniper.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Enemy.h"

#include "Projectile.h"

#include "Tower.h"

#include "Sniper.h"

#include <string>

namespace game\_framework {

Sniper::Sniper() : Tower() {

atk\_speed = 1.5;

MaxSp = 5;

iconX = 400;

cost = 14;

projectile.SetPicNumber(IDB\_SNIPER\_PROJECTILE);

projectile.SetDmg(10);

}

void Sniper::LoadBitmap() {

icon.LoadBitmap(IDB\_EXUSIAI, RGB(255, 255, 255));

char\_l.LoadBitmap(IDB\_EXUISI\_L, RGB(255, 255, 255));

char\_r.LoadBitmap(IDB\_EXUISI\_R, RGB(255, 255, 255));

projectile.LoadBitmap();

Tower::IconReset();

}

void Sniper::OnMove() {

if (IsActive) {

for (int i = 0; i < time.size(); i++) {

if (target[i] != nullptr && IsInRange(pos[i].first, pos[i].second, target[i]->x, target[i]->y) && target[i]->isAlive()) {

if (sp[i] == MaxSp && Attackable(i, 50)) {

Attack(i);

skillHit[i]--;

if (skillHit[i] == 0)

sp[i] = 0;

}

else if (Attackable(i, 1000 / atk\_speed)) {

Attack(i);

sp[i]++;

skillHit[i] = 4;

}

}

}

projectile.OnMove();

}

}

void Sniper::OnShow() {

icon.ShowBitmap();

Tower::OnShow();

}

bool Sniper::IsInRange(int selfX, int selfY, int targetX, int targetY) {

switch (direction)

{

case Up:

if (targetX >= selfX - 48 && targetX <= selfX + 96 && targetY >= selfY - 132 && targetY <= selfY + 60)

return true;

break;

case Down:

if (targetX >= selfX - 48 && targetX <= selfX + 96 && targetY >= selfY + 12 && targetY <= selfY + 204)

return true;

break;

case Left:

if (targetX >= selfX-144 && targetX <= selfX + 48 && targetY >= selfY - 36 && targetY <= selfY + 108)

return true;

break;

case Right:

if (targetX >= selfX && targetX <= selfX + 192 && targetY >= selfY - 36 && targetY <= selfY + 108)

return true;

break;

default:

break;

}

return false;

}

void Sniper::SetData(int x, int y, clock\_t t, int \*cost) {

Tower::SetData(x, y, t, cost);

skillHit.push\_back(4);

}

void Sniper::ClearData() {

Tower::ClearData();

skillHit.clear();

}

}

Guard.h

namespace game\_framework {

class Guard : public Tower {

public:

Guard();

void LoadBitmap();

void OnMove();

void OnShow();

void SetData(int, int, clock\_t, int\*);

void ClearData();

void SetEnemies(vector<Enemy>&);

void Attack(int);

bool IsInRange(int, int, int, int);

private:

vector<int> durationCounter;

vector<Enemy\*> target2;

Projectile projectile2;

};

}

Guard.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Enemy.h"

#include "Projectile.h"

#include "Tower.h"

#include "Guard.h"

#include <string>

namespace game\_framework {

Guard::Guard() : Tower() {

atk\_speed = 1;

MaxSp = 2;

projectile.SetPicNumber(IDB\_LAPPLAND\_PROJECTILE);

projectile2.SetPicNumber(IDB\_LAPPLAND\_PROJECTILE);

iconX = 448;

cost = 18;

projectile.SetDmg(10);

projectile2.SetDmg(10);

MaxBlockCount = 2;

MaxHp = 200;

}

void Guard::LoadBitmap() {

icon.LoadBitmap(IDB\_LAPPLAND, RGB(255, 255, 255));

char\_l.LoadBitmap(IDB\_LAPPLAND\_L, RGB(255, 255, 255));

char\_r.LoadBitmap(IDB\_LAPPLAND\_R, RGB(255, 255, 255));

projectile.LoadBitmap();

projectile2.LoadBitmap();

hpBar.LoadBitmap();

Tower::IconReset();

}

void Guard::OnMove() {

if (IsActive) {

for (int i = 0; i < time.size(); i++) {

if (hp[i] > 0) {

if (target[i] != nullptr && IsInRange(pos[i].first, pos[i].second, target[i]->x, target[i]->y) && target[i]->isAlive()) {

if (sp[i] == MaxSp) {

Attack(i);

sp[i] = 0;

}

else if (Attackable(i, 1000 / atk\_speed)) {

Tower::Attack(i);

sp[i]++;

hp[i] -= 10 \* blockCount[i];

}

}

for (int j = 0; j < blockedEnemies.size(); j++) {

if (!blockedEnemies[j]->isAlive()) {

enemies[j]->isBlocked = false;

blockedEnemies.erase(blockedEnemies.begin() + j);

j--;

blockCount[i]--;

}

}

for (int j = 0; j < enemies.size(); j++) {

if (blockCount[i] < MaxBlockCount && enemies[j]->isAlive() && !enemies[j]->isBlocked && Range(pos[i].first, enemies[j]->x, pos[i].second, enemies[j]->y) < 48) {

enemies[j]->isBlocked = true;

blockedEnemies.push\_back(enemies[j]);

blockCount[i]++;

}

}

}

else {

for (int j = 0; j < blockedEnemies.size(); j++)

blockedEnemies[j]->isBlocked = false;

blockedEnemies.clear();

for (int j = 0; j < blockCount.size(); j++)

blockCount[j] = 0;

}

}

projectile.OnMove();

projectile2.OnMove();

}

}

void Guard::OnShow() {

icon.ShowBitmap();

projectile2.OnShow();

for (int i = 0; i < pos.size(); i++) {

if (hp[i] > 0) {

hpBar.Set(pos[i].first, pos[i].second, hp[i], MaxHp);

hpBar.OnShow();

}

}

Tower::OnShow();

}

void Guard::SetData(int x, int y, clock\_t t, int \*cost) {

target2.push\_back(0);

Tower::SetData(x, y, t, cost);

}

void Guard::ClearData() {

target2.clear();

projectile2.pos.clear();

Tower::ClearData();

}

void Guard::SetEnemies(vector<Enemy>& enemies) {

for (int i = 0; i < pos.size(); i++) {

for (int j = 0; j < enemies.size(); j++) {

if (enemies[j].isAlive() && IsInRange(pos[i].first, pos[i].second, enemies[j].x, enemies[j].y)) {

this->target[i] = &(enemies[j]);

projectile.SetTarget(enemies[j]);

for (int n = j + 1; n < enemies.size(); n++)

if (enemies[n].isAlive() && IsInRange(pos[i].first, pos[i].second, enemies[n].x, enemies[n].y)) {

this->target2[i] = &(enemies[n]);

projectile2.SetTarget(enemies[n]);

break;

}

break;

}

}

}

if (this->enemies.empty())

for (int i = 0; i < enemies.size(); i++)

this->enemies.push\_back(&enemies[i]);

}

void Guard::Attack(int i) {

projectile2.SetPosition(pos[i].first + 24, pos[i].second + 24);

Tower::Attack(i);

}

bool Guard::IsInRange(int selfX, int selfY, int targetX, int targetY) {

switch (int(direction))

{

case 0:

if ((targetX >= selfX - 48 && targetX <= selfX + 96 && targetY >= selfY - 36 && targetY <= selfY + 60) || (targetX >= selfX && targetX <= selfX + 48 && targetY >= selfY - 132 && targetY <= selfY + 60))

return true;

break;

case 1:

if ((targetX >= selfX - 48 && targetX <= selfX + 96 && targetY >= selfY + 12 && targetY <= selfY + 108) || (targetX >= selfX && targetX <= selfX + 48 && targetY >= selfY + 12 && targetY <= selfY + 204))

return true;

break;

case 2:

if ((targetX >= selfX - 48 && targetX <= selfX + 48 && targetY >= selfY - 36 && targetY <= selfY + 108) || (targetX >= selfX - 144 && targetX <= selfX - 48 && targetY >= selfY -12 && targetY <= selfY + 60))

return true;

break;

case 3:

if ((targetX >= selfX && targetX <= selfX + 96 && targetY >= selfY - 36 && targetY <= selfY + 108) || (targetX >= selfX && targetX <= selfX + 192 && targetY >= selfY + 12 && targetY <= selfY + 60))

return true;

break;

default:

break;

}

return false;

}

}

Suportor.h

namespace game\_framework {

class Suportor : public Tower {

public:

Suportor();

void LoadBitmap();

void OnMove();

void OnShow();

bool IsInRange(int, int, int, int);

};

}

Suportor.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Enemy.h"

#include "Projectile.h"

#include "Tower.h"

#include "Suportor.h"

#include <string>

namespace game\_framework {

Suportor::Suportor() : Tower() {

atk\_speed = 1;

MaxSp = 5;

iconX = 496;

cost = 23;

projectile.SetPicNumber(IDB\_PROJECTILE);

projectile.SetDmg(12);

MaxHp = 100;

}

void Suportor::LoadBitmap() {

icon.LoadBitmap(IDB\_WARFARIN, RGB(255, 255, 255));

char\_l.LoadBitmap(IDB\_WARFARIN\_L, RGB(255, 255, 255));

char\_r.LoadBitmap(IDB\_WARFARIN\_R, RGB(255, 255, 255));

projectile.LoadBitmap();

Tower::IconReset();

hpBar.LoadBitmap();

}

void Suportor::OnMove() {

if (IsActive) {

for (int i = 0; i < time.size(); i++) {

if (target[i] != nullptr && IsInRange(pos[i].first, pos[i].second, target[i]->x, target[i]->y) && target[i]->isAlive()) {

if (Attackable(i, 1000 / atk\_speed)) {

Attack(i);

for (int j = 0; j < enemies.size(); j++) {

enemies[j]->speed = enemies[j]->initSpeed;

}

target[i]->speed -= 1;

//OutputDebugString(\_T((to\_string(skill[i].first) + " ").c\_str()));

}

}

}

projectile.OnMove();

}

}

void Suportor::OnShow() {

icon.ShowBitmap();

Tower::OnShow();

}

bool Suportor::IsInRange(int selfX, int selfY, int targetX, int targetY) {

switch (direction)

{

case Up:

if (targetX >= selfX - 48 && targetX <= selfX + 96 && targetY >= selfY - 132 && targetY <= selfY + 60)

return true;

break;

case Down:

if (targetX >= selfX - 48 && targetX <= selfX + 96 && targetY >= selfY + 12 && targetY <= selfY + 204)

return true;

break;

case Left:

if (targetX >= selfX - 144 && targetX <= selfX + 48 && targetY >= selfY - 36 && targetY <= selfY + 108)

return true;

break;

case Right:

if (targetX >= selfX && targetX <= selfX + 192 && targetY >= selfY - 36 && targetY <= selfY + 108)

return true;

break;

default:

break;

}

return false;

}

}

Caster.h

namespace game\_framework {

class Caster : public Tower {

public:

Caster();

void LoadBitmap();

void OnMove();

void OnShow();

bool IsInRange(int, int, int, int);

};

}

Caster.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Enemy.h"

#include "Projectile.h"

#include "Tower.h"

#include "Caster.h"

#include <string>

namespace game\_framework {

Caster::Caster() : Tower() {

atk\_speed = 1;

MaxSp = 5;

iconX = 544;

cost = 21;

projectile.SetPicNumber(IDB\_CASTER\_PROJECTILE);

projectile.SetDmg(10);

}

void Caster::LoadBitmap() {

icon.LoadBitmap(IDB\_EYJAFJALLA, RGB(255, 255, 255));

char\_l.LoadBitmap(IDB\_EYJAFJALLA\_L, RGB(255, 255, 255));

char\_r.LoadBitmap(IDB\_EYJAFJALLA\_R, RGB(255, 255, 255));

projectile.LoadBitmap();

Tower::IconReset();

}

void Caster::OnMove() {

if (IsActive) {

for (int i = 0; i < time.size(); i++) {

if (target[i] != nullptr && IsInRange(pos[i].first, pos[i].second, target[i]->x, target[i]->y) && target[i]->isAlive()) {

if (sp[i] == MaxSp) {

projectile.SetPicNumber(IDB\_CASTER\_PROJECTILE\_SKILL);

Attack(i);

for (int j = 0; j < 5; j++) {

if (Range(enemies[j]->x, target[i]->x, enemies[j]->y, target[i]->y) < 48 \* 2) {

enemies[j]->hp -= 30;

}

}

projectile.SetPicNumber(IDB\_CASTER\_PROJECTILE);

sp[i] = 0;

}

else if (Attackable(i, 1000 / atk\_speed)) {

Attack(i);

sp[i]++;

}

}

}

projectile.OnMove();

}

}

void Caster::OnShow() {

icon.ShowBitmap();

Tower::OnShow();

}

bool Caster::IsInRange(int selfX, int selfY, int targetX, int targetY) {

switch (int(direction))

{

case 0:

if ((targetX >= selfX - 48 && targetX <= selfX + 96 && targetY >= selfY - 84 && targetY <= selfY + 60) || (targetX >= selfX && targetX <= selfX + 48 && targetY >= selfY - 132 && targetY <= selfY - 84))

return true;

break;

case 1:

if ((targetX >= selfX - 48 && targetX <= selfX + 96 && targetY >= selfY + 12 && targetY <= selfY + 156) || (targetX >= selfX && targetX <= selfX + 48 && targetY >= selfY + 156 && targetY <= selfY + 204))

return true;

break;

case 2:

if ((targetX >= selfX - 96 && targetX <= selfX + 48 && targetY >= selfY - 36 && targetY <= selfY + 108) || (targetX >= selfX - 144 && targetX <= selfX - 96 && targetY >= selfY + 12 && targetY <= selfY + 60))

return true;

break;

case 3:

if ((targetX >= selfX && targetX <= selfX + 144 && targetY >= selfY - 36 && targetY <= selfY + 108) || (targetX >= selfX + 144 && targetX <= selfX + 192 && targetY >= selfY + 12 && targetY <= selfY + 48))

return true;

break;

default:

break;

}

return false;

}

}

Defender.h

namespace game\_framework {

class Defender : public Tower {

public:

Defender();

void LoadBitmap();

void OnMove();

void OnShow();

bool IsInRange(int, int, int, int);

};

}

Defender.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Enemy.h"

#include "Projectile.h"

#include "Tower.h"

#include "Defender.h"

#include <string>

namespace game\_framework {

Defender::Defender() : Tower() {

atk\_speed = 1;

MaxSp = 5;

iconX = 592;

cost = 23;

projectile.SetPicNumber(IDB\_PROJECTILE);

projectile.SetDmg(5);

MaxBlockCount = 3;

MaxHp = 500;

}

void Defender::LoadBitmap() {

icon.LoadBitmap(IDB\_HOSHIKUMA, RGB(255, 255, 255));

char\_l.LoadBitmap(IDB\_HOSHIKUMA\_L, RGB(255, 255, 255));

char\_r.LoadBitmap(IDB\_HOSHIKUMA\_R, RGB(255, 255, 255));

projectile.LoadBitmap();

Tower::IconReset();

hpBar.LoadBitmap();

}

void Defender::OnMove() {

if (IsActive) {

for (int i = 0; i < time.size(); i++) {

if (hp[i] > 0) {

if (target[i] != nullptr && IsInRange(pos[i].first, pos[i].second, target[i]->x, target[i]->y) && target[i]->isAlive()) {

if (Attackable(i, 1000 / atk\_speed)) {

Attack(i);

hp[i] -= 10 \* blockCount[i];

for (int j = 0; j < blockedEnemies.size(); j++)

blockedEnemies[j]->hp -= 5;

OutputDebugString(\_T((to\_string(blockCount[i]) + " ").c\_str()));

}

}

for (int j = 0; j < blockedEnemies.size(); j++) {

if (!blockedEnemies[j]->isAlive()) {

enemies[j]->isBlocked = false;

blockedEnemies.erase(blockedEnemies.begin() + j);

j--;

blockCount[i]--;

}

}

for (int j = 0; j < enemies.size(); j++) {

if (blockCount[i] < MaxBlockCount && enemies[j]->isAlive() && !enemies[j]->isBlocked && Range(pos[i].first, enemies[j]->x, pos[i].second, enemies[j]->y) < 48) {

enemies[j]->isBlocked = true;

blockedEnemies.push\_back(enemies[j]);

blockCount[i]++;

}

}

}

else {

for (int j = 0; j < blockedEnemies.size(); j++)

blockedEnemies[j]->isBlocked = false;

blockedEnemies.clear();

for (int j = 0; j < blockCount.size(); j++)

blockCount[j] = 0;

}

}

projectile.OnMove();

}

}

void Defender::OnShow() {

icon.ShowBitmap();

for (int i = 0; i < pos.size(); i++) {

if (hp[i] > 0) {

hpBar.Set(pos[i].first, pos[i].second, hp[i], MaxHp);

hpBar.OnShow();

}

}

Tower::OnShow();

}

bool Defender::IsInRange(int selfX, int selfY, int targetX, int targetY) {

switch (int(direction))

{

case 0:

if (targetX >= selfX && targetX <= selfX + 48 && targetY >= selfY - 36 && targetY <= selfY + 60)

return true;

break;

case 1:

if (targetX >= selfX && targetX <= selfX + 48 && targetY >= selfY + 12 && targetY <= selfY + 108)

return true;

break;

case 2:

if (targetX >= selfX - 48 && targetX <= selfX + 48 && targetY >= selfY + 12 && targetY <= selfY + 60)

return true;

break;

case 3:

if (targetX >= selfX && targetX <= selfX + 96 && targetY >= selfY + 12 && targetY <= selfY + 60)

return true;

break;

}

return false;

}

}

HpBar.h

namespace game\_framework {

class HpBar {

public:

HpBar();

void LoadBitmap();

void Set(int, int, int, int); //設定血條位置和血量顯示

void OnShow();

CMovingBitmap pic1, pic2;

int x, y, hp, maxhp = 100;

};

}

HpBar.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "HpBar.h"

namespace game\_framework {

HpBar::HpBar() {

}

void HpBar::LoadBitmap() {

pic1.LoadBitmap(IDB\_HEALTH1);

pic2.LoadBitmap(IDB\_HEALTH2);

}

void HpBar::Set(int x, int y, int hp, int maxhp) {

this->x = x;

this->y = y;

this->hp = hp;

this->maxhp = maxhp;

}

void HpBar::OnShow() {

int correction = 0;

int percentage = (hp \* 100) / maxhp;

for (int j = 10; j <= 100; j += 10) {

if (j <= percentage) {

pic1.SetTopLeft(x + 4 + correction, y - 8);

pic1.ShowBitmap();

}

else {

pic2.SetTopLeft(x + 4 + correction, y - 8);

pic2.ShowBitmap();

}

correction += 4;

}

}

}

Projectile.h

namespace game\_framework {

class Tower;

class Projectile {

public:

Projectile();

void LoadBitmap();

void OnMove();

void OnShow();

void SetTarget(Enemy&); //設定目標

void SetPosition(int, int); //設定新飛行物的位置

void SetPicNumber(int); //改變顯示的圖片

void SetDmg(int); //改變傷害

void DealDmg(); //對目標造成傷害

Enemy \*target;

Tower \*targetTower;

bool isMedic;

int x, y, picNumber, target\_x, target\_y, targetIndex;

CMovingBitmap pic;

double angle;

vector<pair<int, int>> pos;

private:

double Range(int, int, int, int);

int dmg;

};

}

Projectile.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Enemy.h"

#include "Projectile.h"

#include "Tower.h"

#include <string>

#include <time.h>

namespace game\_framework {

Projectile::Projectile() {

}

void Projectile::LoadBitmap() {

pic.LoadBitmap(picNumber, RGB(255, 255, 255));

}

void Projectile::OnMove() {

for (int i = 0; i < pos.size(); i++) {

if (target) {

double angle = (double)(target->y + 24 - pos[i].second) / (target->x + 24 - pos[i].first);

if (Range(pos[i].first, target->x, pos[i].second, target->y) <= 48 \* 4) {

if (pow(target->x + 24 - pos[i].first, 2) + pow(target->y + 24 - pos[i].second, 2) <= 100) {

DealDmg();

pos.erase(pos.begin() + i);

i -= 2;

}

else if (target->x + 24 == pos[i].first) {

if (target->y + 24 - pos[i].second > 0)

pos[i].second += 10;

else

pos[i].second += -10;

}

else if (target->x + 24 - pos[i].first < 0 && target->y + 24 == pos[i].second) {

pos[i].first += -10;

}

else if (target->x + 24 - pos[i].first > 0 && target->y + 24 == pos[i].second) {

pos[i].first += 10;

}

else if (target->x + 24 - pos[i].first < 0) {

if (target->y + 24 - pos[i].second < 0)

pos[i].second += -10 \* abs(sin(angle));

else

pos[i].second += 10 \* abs(sin(angle));

pos[i].first += -10 \* abs(cos(angle));

}

else {

if (target->y + 24 - pos[i].second < 0)

pos[i].second += -10 \* abs(sin(angle));

else

pos[i].second += 10 \* abs(sin(angle));

pos[i].first += 10 \* abs(cos(angle));

}

}

}

else if (targetTower) {

double angle = (double)(targetTower->pos[targetIndex].second + 24 - pos[i].second) / (targetTower->pos[targetIndex].first + 24 - pos[i].first);

if (Range(pos[i].first, targetTower->pos[targetIndex].first, pos[i].second, targetTower->pos[targetIndex].second) <= 48 \* 4) {

if (pow(targetTower->pos[targetIndex].first + 24 - pos[i].first, 2) + pow(targetTower->pos[targetIndex].second + 24 - pos[i].second, 2) <= 100) {

Heal();

pos.erase(pos.begin() + i);

i -= 2;

}

else if (targetTower->pos[targetIndex].first + 24 == pos[i].first) {

if (targetTower->pos[targetIndex].second + 24 - pos[i].second > 0)

pos[i].second += 10;

else

pos[i].second += -10;

}

else if (targetTower->pos[targetIndex].first + 24 - pos[i].first < 0 && targetTower-> pos[targetIndex].second + 24 == pos[i].second) {

pos[i].first += -10;

}

else if (targetTower->pos[targetIndex].first + 24 - pos[i].first > 0 && targetTower -> pos[targetIndex].second + 24 == pos[i].second) {

pos[i].first += 10;

}

else if (targetTower->pos[targetIndex].first + 24 - pos[i].first < 0) {

if (targetTower->pos[targetIndex].second + 24 - pos[i].second < 0)

pos[i].second += -10 \* abs(sin(angle));

else

pos[i].second += 10 \* abs(sin(angle));

pos[i].first += -10 \* abs(cos(angle));

}

else {

if (targetTower->pos[targetIndex].second + 24 - pos[i].second < 0)

pos[i].second += -10 \* abs(sin(angle));

else

pos[i].second += 10 \* abs(sin(angle));

pos[i].first += 10 \* abs(cos(angle));

}

}

}

}

}

void Projectile::OnShow() {

for (int i = 0; i < pos.size(); i++)

if(target->isAlive() && Range(pos[i].first, target->x, pos[i].second, target->y) <= 48 \* 4 ) {

pic.SetTopLeft(pos[i].first, pos[i].second);

pic.ShowBitmap();

}

}

void Projectile::SetTarget(Enemy& target) {

this->target = &target;

this->targetTower = nullptr;

}

void Projectile::SetPosition(int x, int y) {

pos.push\_back(make\_pair(x, y));

}

void Projectile::DealDmg() {

target->hp -= dmg;

}

double Projectile::Range(int x1,int x2,int y1,int y2) {

return sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2));

}

void Projectile::SetPicNumber(int n) {

picNumber = n;

}

void Projectile::SetDmg(int dmg) {

this->dmg = dmg;

}

}

Cost.h

namespace game\_framework {

class Cost {

public:

Cost();

Cost(int);

void LoadBitmap();

void OnMove();

void OnShow();

void Reset(); //重新開始後要重置

void CheatingMode(); //直接增加費用

CMovingBitmap pic[10];

int initCost, cost, costCounter;

};

}

Cost.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include <string>

#include "Cost.h"

namespace game\_framework {

Cost::Cost() {

this->initCost = 10;

cost = this->initCost;

costCounter = 0;

}

Cost::Cost(int initCost) {

this->initCost = initCost;

cost = this->initCost;

costCounter = 0;

}

void Cost::LoadBitmap() {

int num[10] = { IDB\_0, IDB\_1, IDB\_2, IDB\_3, IDB\_4, IDB\_5, IDB\_6, IDB\_7, IDB\_8, IDB\_9 };

for (int i = 0; i < 10; i++)

pic[i].LoadBitmap(num[i], RGB(0, 0, 0));

}

void Cost::OnMove() {

if (++costCounter >= 30 && cost<99) {

cost++;

costCounter = 0;

}

}

void Cost::OnShow() {

for(int i=0;i<10;i++)

pic[i].SetTopLeft(620, 403);

pic[cost / 10].ShowBitmap();

for(int i=0;i<10;i++)

pic[i].SetTopLeft(630, 403);

pic[cost % 10].ShowBitmap();

}

void Cost::Reset() {

cost = initCost;

costCounter = 0;

}

void Cost::CheatingMode() {

cost = 99;

}

}

LossCounter.h

namespace game\_framework {

class LossCounter {

public:

LossCounter();

LossCounter(int);

void LoadBitmap();

void OnShow();

void Loss();

void CheckEnemies(vector<Enemy>&);

bool CheckLost(); //判斷是否已經輸/贏

bool CheckWin(vector<Enemy>);

int GetLossCount(vector<Enemy>&);

CMovingBitmap pic[10];

int count\_init,count;

};

}

LossCounter.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Enemy.h"

#include "LossCounter.h"

namespace game\_framework {

LossCounter::LossCounter() {

count\_init = count = 3;

}

LossCounter::LossCounter(int c) {

count\_init = count = c;

}

void LossCounter::LoadBitmap() {

int num[10] = { IDB\_0, IDB\_1, IDB\_2, IDB\_3, IDB\_4, IDB\_5, IDB\_6, IDB\_7, IDB\_8, IDB\_9 };

for (int i = 0; i < 10; i++)

pic[i].LoadBitmap(num[i], RGB(0, 0, 0));

}

void LossCounter::OnShow() {

pic[count].SetTopLeft(315, 0);

pic[count].ShowBitmap();

}

void LossCounter::Loss() {

count--;

}

void LossCounter::CheckEnemies(vector<Enemy> &enemies) {

count = count\_init;

for (int i = 0; i < enemies.size(); i++)

if (enemies[i].index == 5 && count > 0) {

count--;

enemies[i].hp = 0;

}

}

bool LossCounter::CheckLost() {

if (count == 0)

return true;

return false;

}

bool LossCounter::CheckWin(vector<Enemy> enemies) {

for (int i = 0; i < enemies.size(); i++) {

if (enemies[i].hp > 0)

break;

if (i == enemies.size() - 1)

return true;

}

return false;

}

int LossCounter::GetLossCount(vector<Enemy> &enemies) {

int lossCount = 0;

for (int i = 0; i < enemies.size(); i++) {

if (enemies[i].index == 5)

lossCount++;

else if (enemies[i].hp > 0)

lossCount++;

}

return lossCount;

}

}

Introduction.h

namespace game\_framework {

class Introduction {

public:

Introduction();

void LoadBitmap();

void OnShow();

void IndexAdd(); //翻頁功能

void IndexMinus();

private:

CMovingBitmap pic[7];

int index;

};

}

Introduction.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Introduction.h"

namespace game\_framework {

Introduction::Introduction() {

index = 0;

}

void Introduction::LoadBitmap() {

pic[0].LoadBitmap(IDB\_INTRODUCTION);

pic[1].LoadBitmap(IDB\_TEXAS\_INTRO);

pic[2].LoadBitmap(IDB\_EXUSIAI\_INTRO);

pic[3].LoadBitmap(IDB\_LAPPLAND\_INTRO);

pic[4].LoadBitmap(IDB\_WARFARIN\_INTRO);

pic[5].LoadBitmap(IDB\_EYJAFJALLA\_INTRO);

pic[6].LoadBitmap(IDB\_HOSHIKUMA\_INTRO);

}

void Introduction::OnShow() {

pic[index].SetTopLeft(0, 0);

pic[index].ShowBitmap();

}

void Introduction::IndexAdd() {

if (index != 6)

index++;

}

void Introduction::IndexMinus() {

if (index != 0)

index--;

}

}