國立臺北科技大學

2020 Spring 資工系物件導向程式實習

期末報告

Age Of Empire II



第4組

108590007 王典爵

108590039 吳柏諭

**目錄**

1. **簡介**
2. 動機 …………………………………………………………. 2
3. 分工 …………………………………………………………. 2
4. **遊戲介紹**
5. 遊戲說明……………………………………………………... 2
6. 遊戲圖形……………………………………………………... 3
7. 遊戲音效……………………………………………………... 4
8. **程式設計**
9. 程式架構……………………………………………………... 5
10. 程式類別……………………………………………………... 6
11. 程式技術……………………………………………………... 8
12. **結語**
13. 問題及解決方法……………………………………………... 8
14. 時間表………………………………………………………....9
15. 貢獻比例………………………………………………………10
16. 自我檢核表……………………………………………………10
17. 收獲……………………………………………………………10
18. 心得、感想……………………………………………………10

**動機:**

原本我們打算要製作Forts這款2D對戰遊戲的，但由於其物理系統等難度過高，所以後來選擇我們的備選方案，世紀帝國二，雖然難度一樣偏高，但剛開始製作時滿懷希望，最後就決定是他了。

**分工:**

108590039吳柏諭負責介面的編寫與整個遊戲流程與世界框架，包括網路連線等服務。108590007 王典爵負責實體的巡路演算法，還有實體的組件模式等設計模式。

**遊戲說明:**

在開始遊戲介面時玩家可以選擇多人遊戲，多人遊戲可以選擇創建伺服器端或連線至伺服器(注意:當選擇創建伺服器時就不能連線至伺服器端，需要重新啟動遊戲才能再次選擇)，選擇連線至伺服器端時輸入伺服器所在ipv4位置可以連線至遊戲(建議Lan端連線，不然容易連線不穩導致不明錯誤)，當連線至伺服器端時兩端需有相同的map.txt才能連線，否則會自動斷開連結。若不想要多人可以按P直接進入遊戲，但此時對手沒有AI。

進入遊戲中後玩家會有3隻村民及1個市政中心，上下左右鍵或點選右下角小地圖可控制地圖移動，左鍵點擊村民或框選村民後底下會列出所選的村民，此時對資源(樹、羊、金礦、石礦)點右鍵，可以對其進行採集，村民會自動採集然後回到市政中心放材料身上資源才會增加，而對敵對的村民或市政中心點右鍵可以控制他前往攻擊該單位，當敵方村民全死光時遊戲就會結束。當點擊己方村民時左下角可以選擇蓋市政中心，當有成功選取到時滑鼠旁會有市政中心的圖案，此時左鍵可以消耗50除肉以外所有資源生成一個未完成的市政中心，而此時右鍵可以取消選取，未完成市政中心需要花費村民蓋才能有所作用。

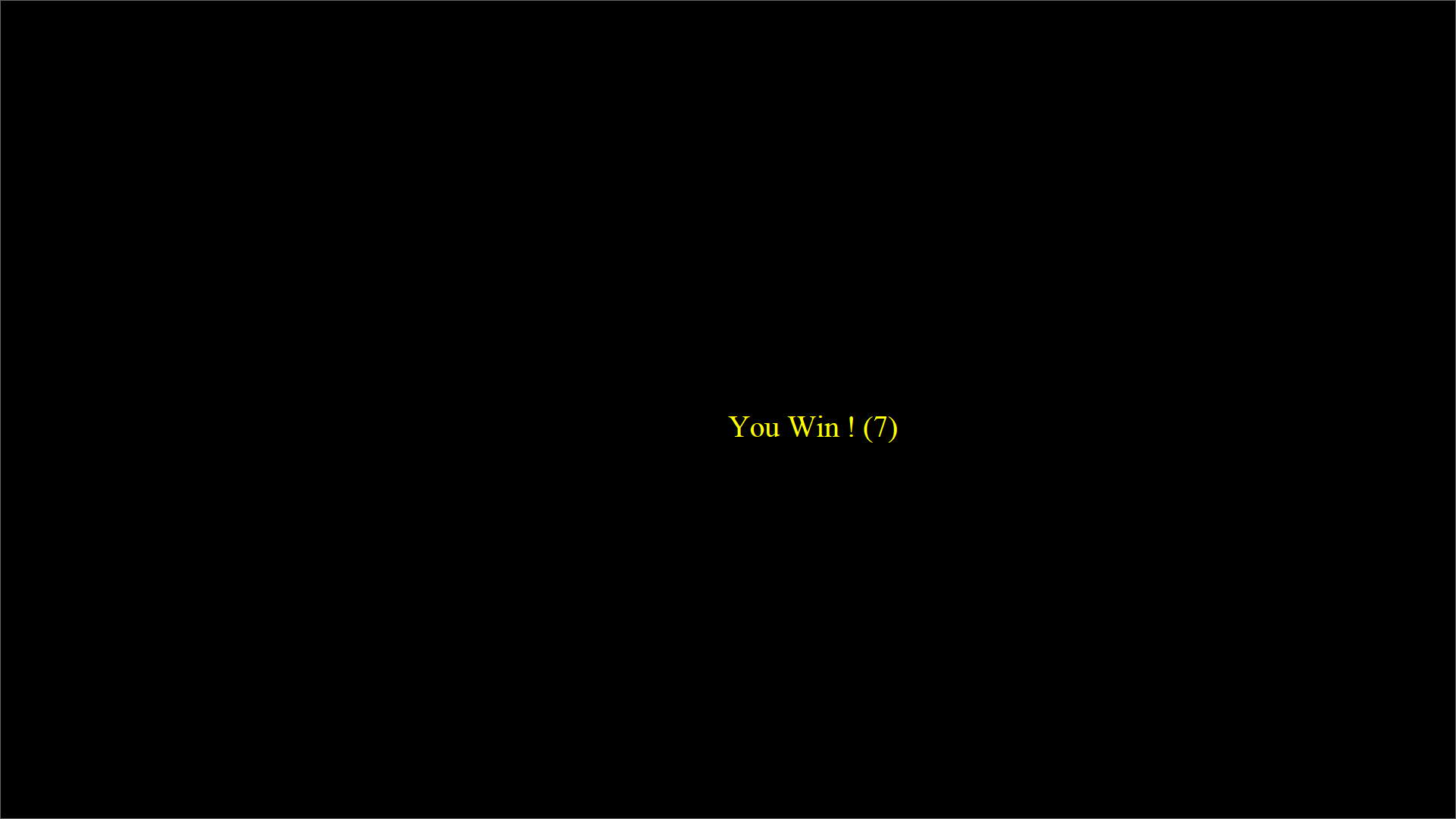
在遊玩時按K可以進入作弊模式，此時所有資源會加500，而市政中心也會直接成完成狀態，再按一下K可以關掉此模式。

在主選端時點選地圖編輯器時可以自由編輯地圖的地形與資源，上下左右鍵可控制地圖移動，編輯完成時按S可以將地圖檔存檔，之後進行遊戲時就會已剛剛編輯完成的地圖開始遊戲(多人遊戲時需有相同地圖)。

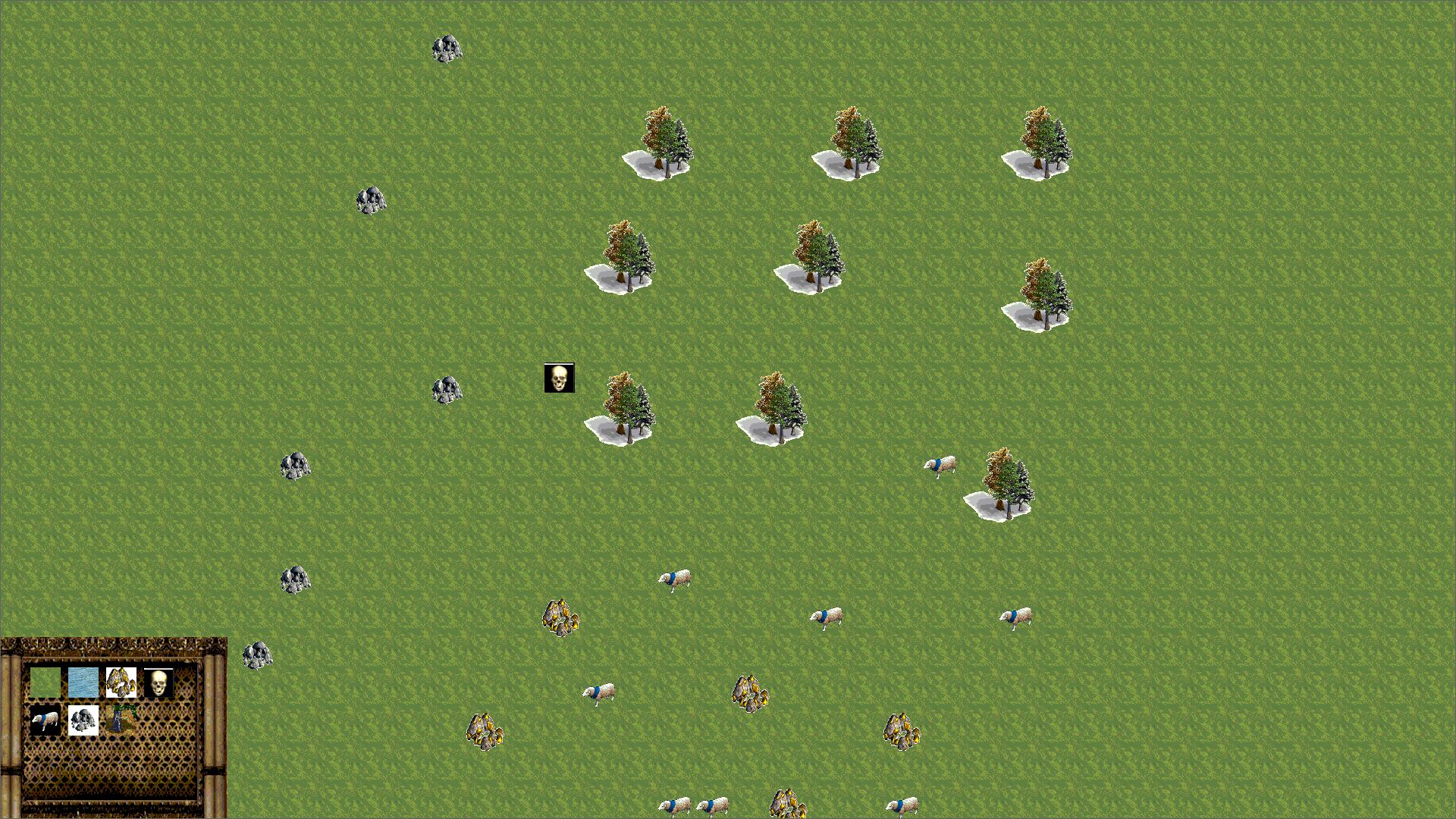
遊戲圖形:



遊玩畫面



獲勝畫面



地圖編輯器畫面

**遊戲音效:**

ding.way

Error.mp3

lake.mp3

MainTheme.mp3

Ntut.mid

SheepSound.mp3

Soundtrack.mp3

TownCenterSound.mp3

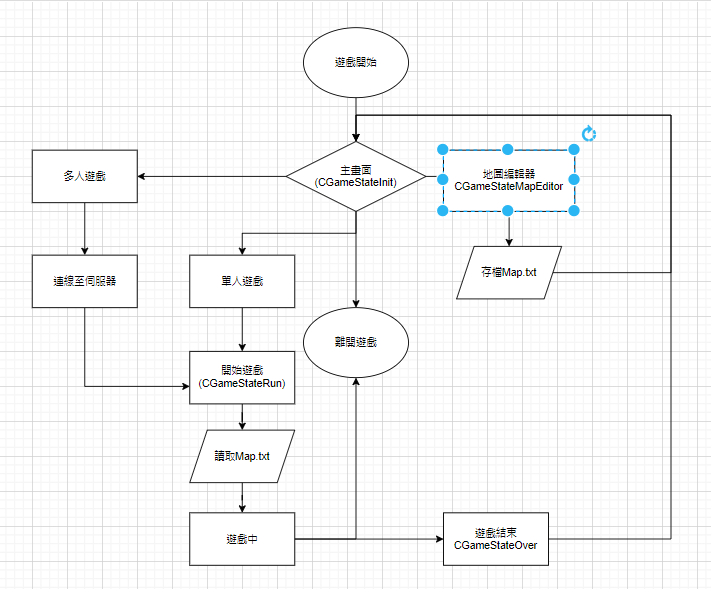
UnitCreation.mp3

VillagerDeathSound.mp3

VillagerSound.mp3

**程式架構:**

World、GUI、NetWork是單例模式，World用來管控整個遊戲中的流程，GUI用來管控遊戲中的介面(frame)，frame負責管控其底下的所有按鈕，NetWork用來管控所有連線。



**程式類別:**

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h行數 | .cpp行數 | 說明 |
| Building\_Button | 15 | 23 | 建造建築物按鈕 |
| Button | 33 | 69 | 所有按鈕基礎結構 |
| eraser\_Button | 15 | 23 | 刪除物件按鈕 |
| Gold\_Button | 15 | 23 | 生成金礦按鈕 |
| Grass\_Button | 15 | 23 | 生成草地按鈕 |
| Sheep\_Button | 15 | 23 | 生成羊按鈕 |
| AboutButton | 13 | 27 | 秀出介紹按鈕 |
| CreateServerButton | 13 | 29 | 創建伺服器按鈕 |
| HelpButton | 13 | 28 | 跳出幫助選單按鈕 |
| JoinServerButton | 13 | 30 | 加入多人遊戲按鈕 |
| JoinServerButton2 | 13 | 28 | 連線至所輸入ip按鈕 |
| LeaveButton | 13 | 28 | 離開遊戲按鈕 |
| MapEditorButton | 13 | 27 | 進入世界編輯器按鈕 |
| MultiGameButton | 13 | 33 | 選擇多人遊戲按鈕 |
| Stone\_Button | 15 | 23 | 生成石頭按鈕 |
| Tree\_Button | 15 | 23 | 生成樹按鈕 |
| Water\_Button | 15 | 23 | 生成水按鈕 |
| AboutFrame | 15 | 28 | 生成介紹介面 |
| EntityDataButtonFrame | 19 | 64 | 遊戲中左下角所含按鈕介面 |
| EntityDataFrame | 20 | 112 | 遊戲中下方實體訊息介面 |
| frame | 87 | 0 | 所有介面基礎結構 |
| HelpFrame | 15 | 31 | 幫助介面 |
| IPFrame | 15 | 37 | 輸入IP介面 |
| MiniMap | 33 | 50 | 小地圖介面 |
| ResourceFrame | 17 | 48 | 左上角資源介面 |
| ServerFrame | 15 | 36 | 多人遊戲介面 |
| StartMenuFrane | 15 | 42 | 主畫面 |
| GUI | 34 | 109 | 控制遊戲中介面的單例 |
| Player | 19 | 0 | 玩家資料檔 |
| GameResource | 48 | 0 | 資源類別檔 |
| ClientSocket | 24 | 65 | 客戶端的連線 |
| ServerSocket | 24 | 68 | 伺服器端的連線 |
| NetWork | 34 | 241 | 管理連線的單例 |
| Attack | 19 | 41 | 攻擊的組件(Component) |
| Entity | 187 | 0 | 所有實體的基礎 |
| EntityFactory | 18 | 68 | 實體的工廠模式 |
| Gatherable | 24 | 0 | 採集的組件 |
| Mine | 48 | 49 | 金礦實體 |
| Navigator | 121 | 605 | 巡路的組件 |
| Sheep | 45 | 57 | 羊實體 |
| Stone | 48 | 49 | 石礦實體 |
| TownCenter | 40 | 89 | 市政中心實體 |
| Tree | 46 | 59 | 樹實體 |
| UnitBase | 124 | 0 | 所有組件的基礎(包括實體) |
| Villager | 105 | 477 | 村民實體 |
| ID | 17 | 14 | 用來管控所有實體ID |
| World | 149 | 816 | 用來管控整個遊戲的單例 |
| 總行數 | 1682 | 3738 |  |

**程式技術:**

此專案使用了設計模式中的單例模式，組件模式，工廠模式，並且使用Astar演算法進行巡路計算，還有使用多線程來增加Astar運算效率，使遊戲不會卡，還有使用MFC中CSocket的連線技術來進行主僕式連線，並且遊戲中使用的網路架構為同步命令模式，為傳輸命令過去使個別電腦各自運算以減少傳輸資料負擔，並且一段時間使用分段同步傳輸來減少使用者網路負擔。

**問題及解決方法:**

使用單例模式時在.cpp檔沒有初始化instance使專案報出”參考了無法解析的錯誤”Bug。一口氣控制很多實體時會因Astar演算法吃很多資源導致遊戲瞬間卡住，等到演算完後才繼續開始動，解決方法是使用多線程來增強運算。野指標問題，解決方法是將刪除的地方統整到一個function，並且確認所有會影響的指標都有正常刪除。使用CArchive進行封包讀取及寫入時無法完整讀取內容，解決方是不使用CArchive改使用stringstream寫成既定格式，之後再轉成char\*傳輸，此優點是易用性跟CArchive不相上下，也沒有CArchive的一些問題，且可以應用在檔案存讀檔上。在建置release時碰到C1010錯誤，解決方法是去專案屬性C/C++ -> 先行編譯標頭檔 ->建立/使用先行編譯標頭 改成 未使用先行編譯標頭檔。

**時間表:**

|  |  |  |
| --- | --- | --- |
| 周次 | 組員: 吳柏諭(時數) | 說明 |
| 1 | 1 | 討論製作遊戲內容 |
| 2 | 1 | Git練習 |
| 3 | 1 | Git+tutorial |
| 4 | 2 | 熟悉遊戲框架 |
| 5 | 6 | 地圖整體顯示及移動 |
| 6 | 15 | 遊戲基礎內容 |
| 7 | 6 | 遊戲基礎內容 |
| 8 | 4 | 介面基處內容 |
| 9 | 5 | 介面基處內容 |
| 10 | 40 | 連線功能 |
| 11 | 40 | 優化連線功能及debug所有memory leak問題，遊戲整體架構完成 |
| 12 | 6 | Debug及新增一些介面 |
| 13 | 4 | Debug不知明例外，還有優化連線內容(使用不同網路同步框架) |
| 14 | 6 | 新增更多實體 |
| 15 | 6 | 地圖編輯器成型 |
| 16 | 3 | 完善地圖編輯器和存讀檔 |
| 17 | 15 | 大趕工完成基礎要求 |
| 總時數 | 161 |  |

**貢獻比例:**

108590039 吳柏諭 60%

108590007 王典爵 40%

**自我檢核表:**

|  |  |  |  |
| --- | --- | --- | --- |
| 項目 | 項目 | 完成否 | 無法完成原因 |
| 1 | 解決Memory leak | █已完成 □未完成 |  |
| 2 | 自訂遊戲Icon | █已完成 □未完成 |  |
| 3 | 全螢幕啟動 | █已完成 □未完成 |  |
| 4 | 有About畫面 | █已完成 □未完成 |  |
| 5 | 初始畫面說明 | █已完成 □未完成 |  |
| 6 | 按鍵及滑鼠用法密技 | █已完成 □未完成 |  |
| 7 | 上傳 setup/apk/source檔 | █已完成 □未完成 |  |
| 8 | Setup檔可正確執行 | █已完成 □未完成 |  |
| 9 | 報告字型、點數、對齊、行距、頁碼等格是正確 | █已完成 □未完成 |  |

**收穫:**

我在這次學會了使用中斷點debug，單例設計模式、組件模式、工廠模式等設計模式，Socket及thread寫法，還有世紀帝國二這款遊戲的網路連線模型及運作原理。

**心得:**

我在這次課程中學到蠻多的，尤其是設計模式，這讓我體會到設計模式的重要性與自己程式能力低下的事實，在學期中雖然有幾周我呈現入迷狀態，回家只想開啟電腦瘋狂寫程式及研究新知識，但大多數時間我並沒有想要開啟寫程式的慾望，也就是每星期三和四開啟程式讓自己有些進展而已，我覺得我需要有更多的熱誠在寫程式上，不然遲早會被淘汰的。

**附錄:**

檔案過多於是只秀出World.cpp所有內容

#include "StdAfx.h"

#include "World.h"

#include "GUI/GUI.h"

#include "socket/NetWork.h"

#include "GUI/Frames/EntityDataFrame.h"

#include "mygame.h"

#include "audio.h"

World\* World::getInstance()

{

return &instance;

}

void World::initMap() {

clearAllEntities();

fstream file; //宣告fstream物件

file.open("Map//Map.txt", ios::in);

if (!file)

TRACE("File can't be opened\n");

char\* pBuf = new char[50000];

CString strData;

stringstream ss;

file.read(pBuf, 50000);

strData = pBuf;

ss << strData;

for (int i = 0; i < 120; i++) {

for (int j = 0; j < 120; j++) {

ss >> map[i][j];

buildingMap[i][j] = 0;

//TRACE("%d, %d : %d\n", j, i, map[i][j]);

}

}

UINT resaurceSize;

ss >> resaurceSize;

TRACE("Size: %d\n", resaurceSize);

for (UINT i = 0; i < resaurceSize; i++) {

int ET;

ss >> ET;//Entity type

int x, y; //Entity Location

ss >> x >> y;

World::getInstance()->spawnResaurce(static\_cast<EntityTypes>(ET), x, y);

}

delete[] pBuf;

file.close();

}

void World::initMapFromNet(stringstream& ss) {

clearAllEntities();

for (int i = 0; i < 120; i++) {

for (int j = 0; j < 120; j++) {

ss >> map[i][j];

buildingMap[i][j] = 0;

//TRACE("%d, %d : %d\n", j, i, map[i][j]);

}

}

UINT resaurceSize;

ss >> resaurceSize;

TRACE("Size: %d\n", resaurceSize);

for (UINT i = 0; i < resaurceSize; i++) {

int ET;

ss >> ET;//Entity type

int x, y; //Entity Location

ss >> x >> y;

World::getInstance()->spawnResaurce(static\_cast<EntityTypes>(ET), x, y);

}

}

World::World() {

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

sx = sy = 50 \* 50; //螢幕座標

}

World::~World() {

clearAllEntities();

TRACE("~World()\n");

}

void World::clearAllEntities() {

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

delete unit[i];

}

unit.clear();

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

delete EnemyUnit[i];

}

EnemyUnit.clear();

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

delete ResaurceUnit[i];

}

ResaurceUnit.clear();

}

int World::getLocationItem(int x, int y) {

int GX = x / 50;

int GY = y / 50;

return map[GY][GX] || buildingMap[GY][GX];

}

void World::OnShow() {

for (int i = -1; i <= SIZE\_X / 50 + 1; i++) { //螢幕顯示40格\*22格邊界預載一格

for (int j = -1; j <= SIZE\_Y / 50 + 1; j++) {

int MX = i \* 50 - sx % 50;//取得螢幕點座標

int MY = j \* 50 - sy % 50;

int GX = i + sx / 50;//取得地圖上的格座標

int GY = j + sy / 50;

if (map[GY][GX] != GRASS && map[GY][GX] != RIVER)

continue;

switch (map[GY][GX])

{

case GRASS:

grass.SetTopLeft(MX, MY);

grass.ShowBitmap();

break;

case RIVER:

river.SetTopLeft(MX, MY);

river.ShowBitmap();

break;

default:

break;

}

}

}

}

void World::onMove() {

if (isMovingDown == true) {

if ((sy + 5) > ((120 \* 50) - (SIZE\_Y))) {

sy = 120 \* 50 - SIZE\_Y;

}

else {

sy += 5;

}

}

if (isMovingUp == true) {

if (sy - 5 < 50) {

sy = 50;

}

else {

sy -= 5;

}

}

if (isMovingLeft == true) {

if (sx - 5 < 50) {

sx = 50;

}

else {

sx -= 5;

}

}

if (isMovingRight == true) {

if ((sx + 5) > (120 \* 50 - SIZE\_X)) {

sx = 120 \* 50 - SIZE\_X;

}

else {

sx += 5;

}

}

}

void World::UnitOnMove() {

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

unit[i]->onMove();

}

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

ResaurceUnit[i]->onMove();

}

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

EnemyUnit[i]->onMove();

}

}

void World::UnitOnShow() {

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

if (isOnScreen(unit[i]->point.x, unit[i]->point.y)) {

unit[i]->onShow(GlobalX2ScreenX(unit[i]->point.x), GlobalY2ScreenY(unit[i]->point.y));

}

}

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

if (isOnScreen(EnemyUnit[i]->point.x, EnemyUnit[i]->point.y)) {

EnemyUnit[i]->onShow(GlobalX2ScreenX(EnemyUnit[i]->point.x), GlobalY2ScreenY(EnemyUnit[i]->point.y));

}

}

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

if (isOnScreen(ResaurceUnit[i]->point.x, ResaurceUnit[i]->point.y)) {

ResaurceUnit[i]->onShow(GlobalX2ScreenX(ResaurceUnit[i]->point.x), GlobalY2ScreenY(ResaurceUnit[i]->point.y));

}

}

if (isSpawningEntity) {

//TRACE("Mouse monitor Location: (%d, %d)\n", mouseLocation.x, mouseLocation.y);

spawningEntityBitmap.SetTopLeft(mouseLocation.x, mouseLocation.y);

spawningEntityBitmap.ShowBitmap();

}

}

bool World::isOnScreen(int x,int y) {

if (x >= sx && x <= sx + SIZE\_X) {

if (y >= sy && y <= sy + SIZE\_Y) {

return true;

}

}

return false;

}

void World::moveScreenUp(bool state) {

isMovingUp = state;

}

void World::moveScreenDown(bool state) {

isMovingDown = state;

}

void World::moveScreenLeft(bool state) {

isMovingLeft = state;

}

void World::moveScreenRight(bool state) {

isMovingRight = state;

}

int World::ScreenX2GlobalX(int x) {

return x + sx;

}

int World::ScreenY2GlobalY(int y) {

return y + sy;

}

CPoint World::Screen2Global(CPoint p) {

return CPoint(p.x + sx, p.y + sy);

}

int World::getScreenX() {

return sx;

}

int World::getScreenY() {

return sy;

}

int World::GlobalX2ScreenX(int x) {

return x - sx;

}

int World::GlobalY2ScreenY(int y) {

return y - sy;

}

void World::LoadBitmap() {

grass.LoadBitmap(IDB\_GRASS);

river.LoadBitmap(IDB\_WaterBig);

}

void World::setScreenLocation(int x, int y) {

sx = x;

if (sx > (120 \* 50 - SIZE\_X)) {

sx = 120 \* 50 - SIZE\_X;

}

sy = y;

if (sy > (120 \* 50 - SIZE\_Y)) {

sy = 120 \* 50 - SIZE\_Y;

}

}

void World::setScreenLocation(CPoint point) {

sx = point.x;

if (sx > (120 \* 50 - SIZE\_X)) {

sx = 120 \* 50 - SIZE\_X;

}

sy = point.y;

if (sy > (120 \* 50 - SIZE\_Y)) {

sy = 120 \* 50 - SIZE\_Y;

}

}

void World::spawn(EntityTypes ET, int x, int y) {

Unit::Entity\* en = entityFactory.SpawnEntity(ET, x, y);

en->playerId = 0; //0是自己1是敵人-1是資源

unit.push\_back(en);

calculatePopulation();

if (isInitingWorld)

return;

stringstream ss;

ss << "spawn" << " ";

ss << ET << " ";

ss << x << " " << y;

NetWork::getInstance()->SendData(ss);

}

void World::spawn(EntityTypes ET, CPoint p) {

Unit::Entity\* en = entityFactory.SpawnEntity(ET, p);

en->playerId = 0; //0是自己1是敵人-1是資源

unit.push\_back(en);

calculatePopulation();

if (isInitingWorld)

return;

stringstream ss;

ss << "spawn" << " ";

ss << ET << " ";

ss << p.x << " " << p.y;

NetWork::getInstance()->SendData(ss);

}

void World::spawnEnemy(EntityTypes ET, int x, int y) {

Unit::Entity\* en = entityFactory.SpawnEntity(ET, x, y);

en->playerId = 1; //0是自己1是敵人-1是資源

EnemyUnit.push\_back(en);

}

void World::spawnEnemy(EntityTypes ET, CPoint p) {

Unit::Entity\* en = entityFactory.SpawnEntity(ET, p);

en->playerId = 1; //0是自己1是敵人-1是資源

EnemyUnit.push\_back(en);

}

void World::spawnResaurce(EntityTypes ET, int x, int y) {

int tileX = x / 50;

int tileY = y / 50;

if (buildingMap[tileY][tileX] == 1)

return;

Unit::Entity\* en = entityFactory.SpawnEntity(ET, tileX \* 50, tileY \* 50);

en->playerId = -1; //0是自己1是敵人-1是資源

ResaurceUnit.push\_back(en);

}

void World::spawnResaurce(EntityTypes ET, CPoint p) {

int tileX = p.x / 50;

int tileY = p.y / 50;

if (buildingMap[tileY][tileX] == 1)

return;

Unit::Entity\* en = entityFactory.SpawnEntity(ET, tileX \* 50, tileY \* 50);

en->playerId = -1; //0是自己1是敵人-1是資源

ResaurceUnit.push\_back(en); //0是自己1是敵人-1是資源

}

void World::calculatePopulation() {

int total = 0;

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

if (typeid(\*unit[i]) == typeid(Unit::Villager)) {

total++;

}

}

player.population = total;

}

vector<Unit::Entity\*> World::listAllEntityInRange(CPoint p1, CPoint p2) {

vector<Unit::Entity\*> output;

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

if ((unit[i]->point.x >= p1.x && unit[i]->point.x <= p2.x && unit[i]->point.y >= p1.y && unit[i]->point.y <= p2.y) || (unit[i]->point.x >= p2.x && unit[i]->point.x <= p1.x && unit[i]->point.y >= p2.y && unit[i]->point.y <= p1.y) ||

(unit[i]->point.x >= p2.x && unit[i]->point.x <= p1.x && unit[i]->point.y >= p1.y && unit[i]->point.y <= p2.y) || (unit[i]->point.x >= p1.x && unit[i]->point.x <= p2.x && unit[i]->point.y >= p2.y && unit[i]->point.y <= p1.y)) {

output.push\_back(unit[i]);

//TRACE("Entity: %d\n", i);

}

}

return output;

}

Unit::Entity\* World::getNearestEntity(CPoint point) {

Unit::Entity\* output = NULL;

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

if (unit[i]->HitBox.PtInRect(point)){

output = unit[i];

return output;

}

}

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

if (EnemyUnit[i]->HitBox.PtInRect(point)) {

output = EnemyUnit[i];

return output;

}

}

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

if (ResaurceUnit[i]->HitBox.PtInRect(point)) {

output = ResaurceUnit[i];

return output;

}

}

return output;

}

void World::moveEntityToLocation(vector<Unit::Entity\*> allEntity, CPoint p) {

if (allEntity.size() < 1)

return;

if (allEntity[0]->playerId != 0)

return;

stringstream ss;// command = settarget <amount> <id id id ... id> x y

ss << "settarget" << " ";

ss << allEntity.size() << " ";

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

{

ss << allEntity[i]->ID << " ";

allEntity[i]->SetTarget(p, allEntity);

}

ss << p.x << " " << p.y;

NetWork::getInstance()->SendData(ss);

}

void World::spawningEntity(int bitmap) {

CMovingBitmap MB;

MB.LoadBitmap(bitmap, RGB(255, 255, 255));

spawningEntityBitmap = MB;

switch (bitmap){

case IDB\_VILLAGER000:

spawningEntityType = EntityTypes::Villager;

isEditingMap = 0;

break;

case IDB\_TOWNCENTER\_ICON:

spawningEntityType = EntityTypes::TownCenter;

isEditingMap = 0;

break;

case IDB\_GOLD:

spawningEntityType = EntityTypes::GoldMine;

isEditingMap = 0;

break;

case IDB\_STONE:

spawningEntityType = EntityTypes::Stone;

isEditingMap = 0;

break;

case IDB\_SHEEP:

spawningEntityType = EntityTypes::Sheep;

isEditingMap = 0;

break;

case IDB\_TREE:

spawningEntityType = EntityTypes::Tree;

isEditingMap = 0;

break;

case IDB\_GRASS:

isEditingMap = 1;

break;

case IDB\_WaterBig:

isEditingMap = 2;

break;

case IDB\_DESTROY\_BUTTON:

isEditingMap = 3;

break;

default:

break;

}

isSpawningEntity = true;

}

void World::LoadEnemyFromStringStream(int amount, stringstream& ss) {

int size = EnemyUnit.size();

//TRACE("Size: %d\n", amount);

if (size != amount)

return;

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

string erase;

int ET;

ss >> erase >> ET;

if (ET == static\_cast<int>(EntityTypes::Villager)) {

dynamic\_cast<Unit::Villager\*>(EnemyUnit.at(i))->deSerialize(ss);

}

else if (ET == static\_cast<int>(EntityTypes::TownCenter)) {

dynamic\_cast<Unit::TownCenter\*>(EnemyUnit.at(i))->deSerialize(ss);

}

else { //略過出問題的資料

TRACE("ERROR on update Enemy");

TRACE("ET = %d, %s\n", ET, ss.str().c\_str());

ss >> erase;

while (erase != "End") {

ss >> erase;

}

break;

}

}

}

void World::LoadUnitFromStringStream(int amount, stringstream& ss) {

int size = unit.size();

TRACE("Size: %d %d\n", amount, size);

if (size != amount)

return;

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

string erase;

int ET;

ss >> erase;

TRACE("erase: %s\n", erase.c\_str());

ss >> ET;

TRACE("ER: %d\n", ET);

if (ET == static\_cast<int>(EntityTypes::Villager)) {

dynamic\_cast<Unit::Villager\*>(unit.at(i))->deSerialize(ss);

}

else if (ET == static\_cast<int>(EntityTypes::TownCenter)) {

dynamic\_cast<Unit::TownCenter\*>(unit.at(i))->deSerialize(ss);

}

else {//略過出問題的資料

TRACE("ERROR on update Unit");

ss >> erase;

TRACE("ET = %d, %s\n", ET, erase.c\_str());

ss >> erase;

TRACE("erase: %s\n", erase.c\_str());

break;

}

}

}

void World::LoadResourceFromStringStream(int amount, stringstream& ss) {

int size = ResaurceUnit.size();

if (size != amount)

return;

//TRACE("Size: %d\n", amount);

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

string erase;

int ET;

ss >> erase >> ET;

if (ET == static\_cast<int>(EntityTypes::GoldMine)) {

dynamic\_cast<Unit::Mine\*>(ResaurceUnit.at(i))->deSerialize(ss);

}

else if (ET == static\_cast<int>(EntityTypes::Stone)) {

dynamic\_cast<Unit::Stone\*>(ResaurceUnit.at(i))->deSerialize(ss);

}

else if (ET == static\_cast<int>(EntityTypes::Sheep)) {

dynamic\_cast<Unit::Sheep\*>(ResaurceUnit.at(i))->deSerialize(ss);

}

else if (ET == static\_cast<int>(EntityTypes::Tree)) {

dynamic\_cast<Unit::Tree\*>(ResaurceUnit.at(i))->deSerialize(ss);

}

else { //略過出問題的資料

TRACE("ERROR on update Resource");

TRACE("ET = %d, %s\n", ET, ss.str().c\_str());

ss >> erase;

while (erase != "End") {

ss >> erase;

}

break;

}

}

}

void World::packUnit(vector<Unit::Entity\*> entitys, int type) {

stringstream cmd2;

if(type == 1)

cmd2 << "UpdateEntity1 ";

else if(type == 2)

cmd2 << "UpdateEntity2 ";

else if (type ==3)

cmd2 << "UpdateEntity3 ";

cmd2 << entitys.size() << " ";

//TRACE("%s\n", cmd2.str().c\_str());

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

EntityTypes ET = entitys.at(i)->entityType;

switch (ET) {

case EntityTypes::Villager:

dynamic\_cast<Unit::Villager\*>(entitys.at(i))->Serialize(cmd2);

break;

case EntityTypes::GoldMine:

dynamic\_cast<Unit::Mine\*>(entitys.at(i))->Serialize(cmd2);

break;

case EntityTypes::Sheep:

dynamic\_cast<Unit::Sheep\*>(entitys.at(i))->Serialize(cmd2);

break;

case EntityTypes::Stone:

dynamic\_cast<Unit::Stone\*>(entitys.at(i))->Serialize(cmd2);

break;

case EntityTypes::Tree:

dynamic\_cast<Unit::Tree\*>(entitys.at(i))->Serialize(cmd2);

break;

case EntityTypes::TownCenter:

dynamic\_cast<Unit::TownCenter\*>(entitys.at(i))->Serialize(cmd2);

break;

}

}

NetWork::getInstance()->SendData(cmd2);

}

Unit::Entity\* World::getEntityByID(unsigned int ID) {

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

if (unit[i]->ID == ID)

return unit[i];

}

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

if (EnemyUnit[i]->ID == ID)

return EnemyUnit[i];

}

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

if (ResaurceUnit[i]->ID == ID)

return ResaurceUnit[i];

}

return NULL;

}

vector<Unit::Entity\*> World::getEntityByID(vector<UINT> id) {

vector<Unit::Entity\*> output;

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

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

if (unit[i]->ID == id[j]) {

output.push\_back(unit[i]);

}

}

}

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

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

if (EnemyUnit[i]->ID == id[j]) {

output.push\_back(EnemyUnit[i]);

}

}

}

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

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

if (ResaurceUnit[i]->ID == id[j]) {

output.push\_back(ResaurceUnit[i]);

}

}

}

return output;

}

void World::killByID(UINT ID) {

if (GUI::getInstance()->frames.size() > 2) {

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

if (LE[i]->ID == ID) {

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

}

}

dynamic\_cast<EntityDataFrame\*>(GUI::getInstance()->frames.at(2))->loadEntitysBitmap(World::getInstance()->LE);

}

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

if (unit[i]->ID == ID) {

if (this->isOnScreen(unit[i]->point.x, unit[i]->point.y) && dynamic\_cast<Unit::Villager\*>(unit[i])) {

CAudio::Instance()->Play(AUDIO\_VILLAGERDEATH, false);

}

delete unit[i];

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

calculatePopulation();

stringstream command;

command << "killEntity" << " ";

command << ID;

NetWork::getInstance()->SendData(command);

calculatePopulation();

return;

}

}

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

if (EnemyUnit[i]->ID == ID) {

if (this->isOnScreen(EnemyUnit[i]->point.x, EnemyUnit[i]->point.y) && dynamic\_cast<Unit::Villager\*>(EnemyUnit[i])) {

CAudio::Instance()->Play(AUDIO\_VILLAGERDEATH, false);

}

delete EnemyUnit[i];

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

if (NetWork::getInstance()->isConnectedToClient) {

stringstream command;

command << "killEntity" << " ";

command << ID;

NetWork::getInstance()->SendData(command);

return;

}

else {

int total = 0;

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

if (typeid(\*EnemyUnit[j]) == typeid(Unit::Villager)) {

total++;

}

}

if (total == 0) {

World::getInstance()->isWin = true;

game\_framework::CGame::Instance()->SetGameState(GAME\_STATES::GAME\_STATE\_OVER);

}

return;

}

}

}

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

if (ResaurceUnit[i]->ID == ID) {

delete ResaurceUnit[i];

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

stringstream command;

command << "killEntity" << " ";

command << ID;

NetWork::getInstance()->SendData(command);

return;

}

}

TRACE("ID: %d Not Found\n", ID);

}

void World::initWorld() {

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

delete unit[i];

}

unit.clear();

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

delete EnemyUnit[i];

}

EnemyUnit.clear();

World::getInstance()->player.food = 100;

World::getInstance()->player.gold = 50;

World::getInstance()->player.stone = 50;

World::getInstance()->player.wood = 50;

if (NetWork::getInstance()->isServer()) {//順序要一樣確保初始ID相同

World::getInstance()->cheaterMode = true;

setScreenLocation(1000, 2500);

spawn(EntityTypes::Villager, 1500, 2800);

spawn(EntityTypes::Villager, 1600, 2900);

spawn(EntityTypes::Villager, 1500, 2950);

spawn(EntityTypes::TownCenter, 1600, 3000);

spawnEnemy(EntityTypes::Villager, 4000, 2800);

spawnEnemy(EntityTypes::Villager, 4100, 2900);

spawnEnemy(EntityTypes::Villager, 4000, 2950);

spawnEnemy(EntityTypes::TownCenter, 4100, 3000);

World::getInstance()->cheaterMode = false;

}

else {//順序要一樣確保初始ID相同

World::getInstance()->cheaterMode = true;

setScreenLocation(3500, 2500);

spawnEnemy(EntityTypes::Villager, 1500, 2800);

spawnEnemy(EntityTypes::Villager, 1600, 2900);

spawnEnemy(EntityTypes::Villager, 1500, 2950);

spawnEnemy(EntityTypes::TownCenter, 1600, 3000);

spawn(EntityTypes::Villager, 4000, 2800);

spawn(EntityTypes::Villager, 4100, 2900);

spawn(EntityTypes::Villager, 4000, 2950);

spawn(EntityTypes::TownCenter, 4100, 3000);

World::getInstance()->cheaterMode = false;

}

isInitingWorld = false;

}

void World::setMap(CPoint p, int type) {

int y = p.y / 50;

int x = p.x / 50;

map[y][x] = type;

}

void World::save() {

fstream file; //宣告fstream物件

stringstream contain;

file.open("Map//Map.txt", ios::out | ios::trunc);

if (!file)

TRACE("File can't be opened\n");

for (int i = 0; i < 120; i++) {

for (int j = 0; j < 120; j++) {

contain << map[i][j] << " ";

}

}

contain << ResaurceUnit.size() << " ";

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

contain << ResaurceUnit.at(i)->entityType << " " << ResaurceUnit.at(i)->point.x << " " << ResaurceUnit.at(i)->point.y << " ";

}

string str = contain.str();

char\* output = new char[str.length() + 1];

strcpy(output, str.c\_str());

file.write(output, str.length() + 1);

TRACE("Saving done\n");

file.close();

delete[] output;

}

UINT World::checkMap() {

fstream file; //宣告fstream物件

file.open("Map//Map.txt", ios::in);

if (!file)

TRACE("File can't be opened\n");

char\* pBuf = new char[50000];

CString strData;

stringstream ss;

file.read(pBuf, 50000);

strData = pBuf;

stringstream test;

test << strData;

UINT total = 0;

for (int i = 0; i < 120; i++) {

for (int j = 0; j < 120; j++) {

int temp = 0;

test >> temp;

total += temp;

//TRACE("%d, %d : %d\n", j, i, map[i][j]);

}

}

UINT resaurceSize;

test >> resaurceSize;

for (UINT i = 0; i < resaurceSize; i++) {

int ET;

test >> ET;//Entity type

int x, y; //Entity Location

test >> x >> y;

total += ET + x + y;

}

delete [] pBuf;

return total;

}

World World::instance;