中原大學資訊工程學系

專題實驗 成果報告

大型多人線上遊戲製作器

MMORPG Maker

指導老師：吳宜鴻　助理教授

專題成員：王澤浩　10027129

本專題程式碼收錄於以此

https://github.com/grass0916/Stardust-Script

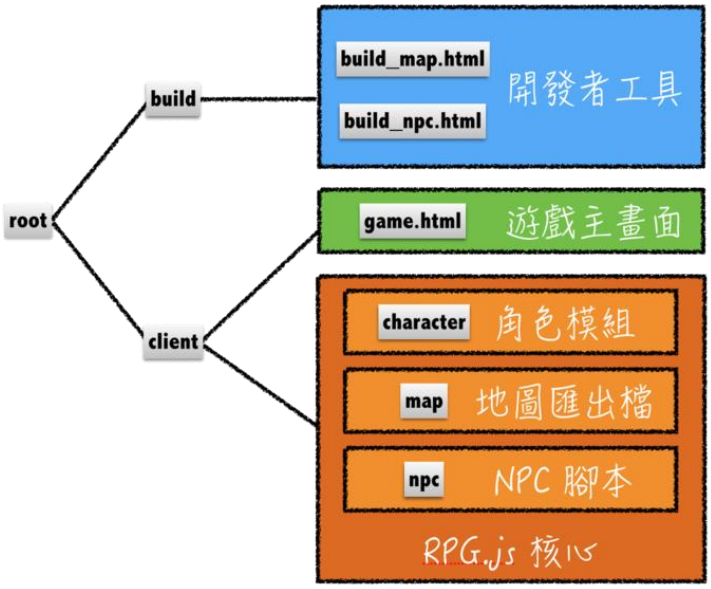
目錄

一、專題摘要

（一）前言

本專題將實作出一系統可自由建置基於網頁平台的多人線上角色扮演遊戲編輯器，使用者可以使用設計好的一系列地圖編輯器、NPC 編輯器來打造專屬的遊戲內容，並利用 Node.js 快速架設遊戲伺服器。

為了建構出擴充性高的遊戲環境，本專題 MMORPG Maker 將會開發以下內容「地圖編輯器（Tiled Map Editor）、非玩家控制角色編輯器（NPC Maker）、主要遊戲框架（Main Game Framework）」，並使用 HTML5 的 Canvas 作為呈現平台。不論是桌上型電腦、行動裝置，使用可支援 HTML5 的瀏覽器將可進行遊戲藉以達到跨平台的目標。專題整體架構如圖一所示。



圖一、系統簡易架構圖。

本專題使用的遊戲角色、物件圖片版權均屬韓國遊戲公司 Gravity 所有，僅用於此專題作為學術用途，亦無任何商業營利等行為。

二、專題研究動機背景與目的

（一）專題研究動機背景

在修習專題實驗、專題研究初期，程式原為在韓國線上遊戲「仙境傳說 Ragnarok Online」的伺服器模擬器 rAthena [1] 上做延伸，但如此一來未來的延伸性將會被限縮在此遊戲框架上。為了希冀未來的程式碼能夠讓更多人擴充的條件下，後續的專題將使用 JavaScript 作為開發語言，並以開放式原始碼（Open Source）的方式開源到網路上。

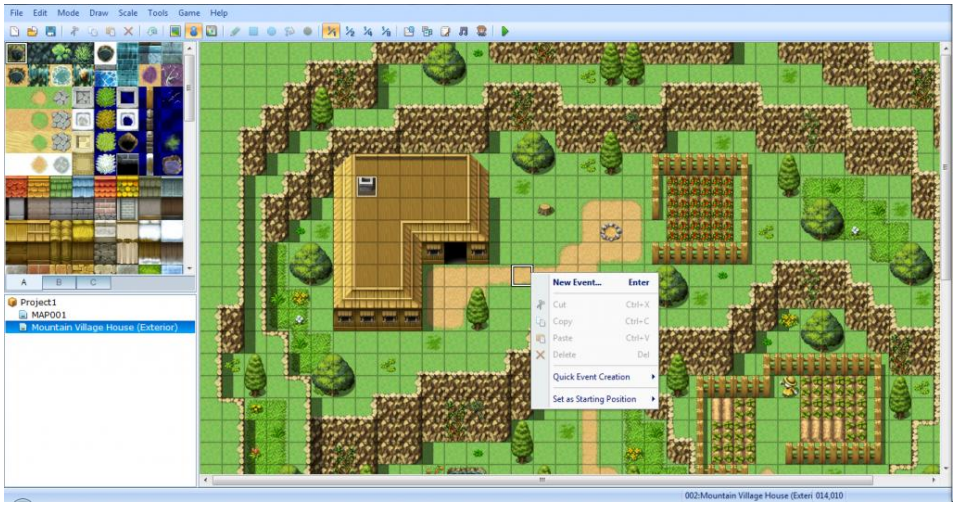
（二）專題研究目的

遊戲製作大師（RPG Maker）[2] 是一套可以打造單機角色扮演遊戲的製作器，藉由此開發出的遊戲卻僅限於 PC 上執行，且需要多人連線還需要額外搭載 socket 連線程式才可達到連線功能。因此本專題將結合幾項現有專案程式的特色，嘗試將其結合後製作出一套擴充性高、跨平台以及即時性多人連線等特色的 2.5D [3] 線上遊戲製作器。

三、個案研究

（一）RPG Maker

RPG Maker 是一套為了能夠讓不懂得程式邏輯的一般使用者也能夠製作出角色扮演遊戲的工具。它是由三個主要元素所構成的，分別為「地圖編輯器」、「事件編輯器」、「戰鬥編輯器」。在本次專題當中，將會以 RPG Maker 的地圖編輯器作為效仿、改良的參考原型。



圖二、RPG Maker 的地圖編輯器畫面。[4]

（二）rAthena

rAthena 是韓國知名線上遊戲「仙境傳說 Ragnarok」的伺服器模擬程式。rAthena 是原 eAthena 專案中衍生出的眾多分支之一。在 rAthena 中可以使用提供的 script command 來撰寫遊戲內的事件。本次專題將會參考過去撰寫 rAthena scripting 的經驗來設計 NPC 的事件處理。

（三）Scratch

Scratch [5] 是由麻省理工學院 Media Lab 所開發的程式語言暨開發平台，其設計之使用者介面採用拖放方塊作組合後便能完成整支程式，而不同的方塊象徵著一程式模件能夠控制特定的角色或是背景。在此次專題將會以 Scratch 的圖形化程式作為仿效對象。

圖三、Scratch 的操作畫面。[6]

四、專題實作方法

（一）地圖編輯器（Tiled Map Editor）

因應輸出之遊戲將採用 2.5D 的瓦磚地圖，地圖編輯器將以相同之開發目標為基底。地圖建造功能劃分為「地圖紋理（Texture）、行走設定（Walkable）、建築物件（Object）、燈光特效（Light）、環境音效（Sound）」共五大部分，並將地圖資料輸出成 JSON 資料交換語言（JavaScript Object Notation）格式供未來遊戲載入、二度編輯時使用。而第五部分環境音效目前歸類於未來計劃之中。

圖四、地圖編輯器的主要界面視覺圖。

於第五章中會詳細說明地圖編輯器的部份。並依照下述條列進行介紹。

1. 地圖的資料結構（Data structure of map）
2. 地圖編輯器的設計（Design of map editor）
    a. 素材選取區（Material box）
    b. 地圖預覽區（Preview box）
3. 地圖紋理（Texture）
    a. 使用者介面（User interface in texture）
    b. 紋理資訊的資料結構（Data Structure in texture）
4. 行走設定（Walkable）
    a. 使用者界面（User interface in walkable）
    b. 行走資訊的資料結構（Data Structure in walkable）
5. 建築物件（Object）

a. 使用者界面（User interface in object）

b. 建築物件資訊的資料結構（Data Structure in object）

c. 建築物件工具箱（Tool box）

6. 燈光特效（Light）

a. 使用者界面（User interface in light）

b. 軌跡公式（Animation function）

7. 地圖輸出、輸入（Export & import）

a. 新開地圖（New map）

b. 輸出地圖（Export a map）

c. 輸入地圖（Import a map）

（二）主要遊戲框架（Main Game Framework）

此一部分將會從線上遊戲環境相關環節逐步進行介紹。從地圖環境建置，如何將上一章的地圖輸出檔呈現至畫面當中、為了呈現 2.5D 的遊戲畫面會需要做的處理以及瓦磚型遊戲會遇到的路線問題。接下來是角色設定，角色會具備那些遊戲上的資訊、角色的模組該如何呈現出來。以及高度擴充性的 NPC 腳本如何實作，將一個角色級物件的動作給模組化，讓後續開發者僅需一行 OnWalk(grid) 即可完成 NPC 的行走動作。接著是多人連線遊玩需要解決的 socket 問題，最後必須將伺服器的重要資訊、角色資訊連接至資料庫中儲存。

圖五、實際遊戲畫面。

於第六章中會詳細說明遊戲框架的部份。並依照下述條列進行介紹。

1. 遊戲架構規劃（Structure plan）
2. 精靈動畫（Sprite）
    a. 角色級玩家物件 Sprite（Sprite of player characters）
    b. 角色級非玩家物件 Sprite（Sprite of non-player characters）
    c. 角色級怪物物件 Sprite（Sprite of monsters）
3. 玩家角色（Player character）
    a. 物件 Character 的結構（Structure of Character）
    b. 物件事件監聽（OnActive）
    c. 物件移動（OnMove）
    d. 物件行走（OnWalk）
    e. 物件 sprite 動畫（OnDirection）
    f. 物件對話泡泡框（OnTalk & OffTalk）
4. 非玩家角色（Non-Player character）
    a. 腳本撰寫（Scripting）
    b. 同步與非同步問題（Asynchronous problem）
    c. 新增 NPC（Add NPC）
5. 路徑規劃（Route planning）
    a. 最短路徑 A*演算法（Shortest Path, A-Star algorithm）
6. 多人連線遊戲（Multi-player gaming）
    a. Web socket - Socket.io
7. 連接資料庫（Connect database）

五、地圖編輯器

（一）地圖的資料結構（Data structure of map）

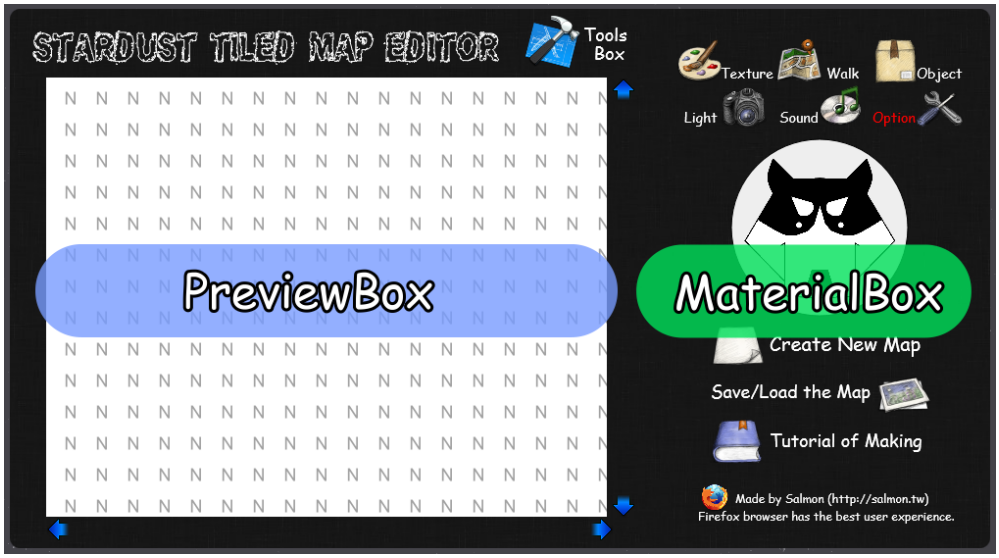
地圖格式會儲存的資訊如下述：

變數名稱	中文簡述	變數意義解釋
tilePixel	單一瓦磚像素	預設值為 32 pixel 的正方形，允許使用 2 的指數之值。
name	地圖名稱	地圖的唯一名稱。
height	地圖高度格數	總列數（rows）。
length	地圖長度格數	總行數（columns）。
tileData	瓦磚資訊陣列	以每列（row）的順序記錄成陣列格式。
objectData	建築物件陣列	依照物件的前後順序（order）作儲存上的排序。
lightData	燈光效果物件陣列	
soundData	環境音效物件陣列	

表一、地圖的儲存資料結構。

（二）地圖編輯器的設計（Design of map editor）

地圖編輯器主要分為地圖預覽區（Preview Box）、素材選取區（Material Box）。



圖六、地圖編輯器主要劃分為兩部分規劃。

在 build_map.html 中，先行匯入建置地圖的 JavaScript files（src: build\src\build\map），接著宣告素材選取區為物件 material 後，並將其傳入地圖預覽區 preview 中取用，藉此取得當前素材選取區的所在頁面、所選物件供其設置。MaterialBox 與 PreviewBox 將以函數之建構子（Constructor）採用「Object.prototype.constructor」的模式撰寫。為了能夠使編輯地圖的過程會有較好的使用者體驗，繪製編輯介面時引入了部分的轉場動畫特效以利畫面流暢。

build\src\build\map\material_box.js	build\src\build\map\preview_box.js

function MaterialBox() {
this.OnCreate() ;
return this ;
} // MaterialBox()
	function PreviewBox(material) {
	this.OnCreate(material) ;
	return this ;
	} // PreviewBox()

build\build_map.html
var material = new MaterialBox() ; var preview = new PreviewBox(material) ;

圖七、MaterialBox 及 PreviewBox 的宣告方式。

1. 素材選取區（Material box）

素材選取區的程式碼收錄在 build\src\build\map\material_box.js 中。在 Material Box 建構子當中依照不同功能性質主要了七個部分，分別為：

物件性質	建構子	主要功能
MaterialBox.prototype.	OnCreate	繪製使用介面，並且導入其它素材功能。
	OnTexture	地圖紋理素材選擇。
	OnWalkable	設置可行走、不可行走。
	OnObject	建築物件素材選擇。
	OnLight	燈光特效素材選擇。
	OnSound	環境音效素材選擇。
	OnOption	新開、儲存、載入地圖。

表二、MaterialBox 的函數建構子。

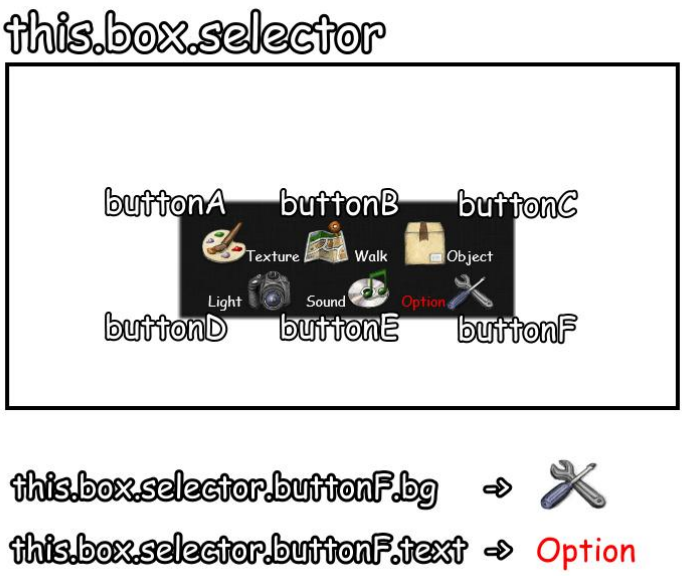
其中在 OnCreate 繪製介面當中，為了達到方便管理每個繪圖元素，需要有物件導向概念

會方便處理，於是使用了 CreateJS 中的 Container()將物件間層層包覆。this.box 為 Canvas 本身，this.box.selector 為六項功能的選擇按鈕，this.box.selector.buttonA 至 buttonF 將是不同按鈕中的圖片及文字元素。且其中每一個繪圖元素都可設定 x 座標及 y 座標，並在有父子關係下可以用 tree 的資料結構方式畫出其關係圖。

build\src\build\map\material_box.js

```
MaterialBox.prototype.OnCreate = function() {
        var that = this ;
        // Material box.
        this.box = new createjs.Container() ;
        this.box.x = 650, this.0062ox.y = 0 ;
        // Material selector container.
        this.box.selector = new createjs.Container() ;
        this.box.selector.x = 25, this.box.selector.y = 25 ;
        this.box.selector.statusPage = 0 ;
        // Material list container.
        this.box.list = new createjs.Container() ;
        this.box.list.x = 30, this.box.list.y = 130 ;
        this.box.addChild( this.box.selector, this.box.list ) ;
        …
} // OnCreate()
```

圖八、於 MaterialBox.js 中節錄的 Container 使用方式。



圖九、Container 中的 buttonF 父子關係。

2. 地圖預覽區（Preview box）

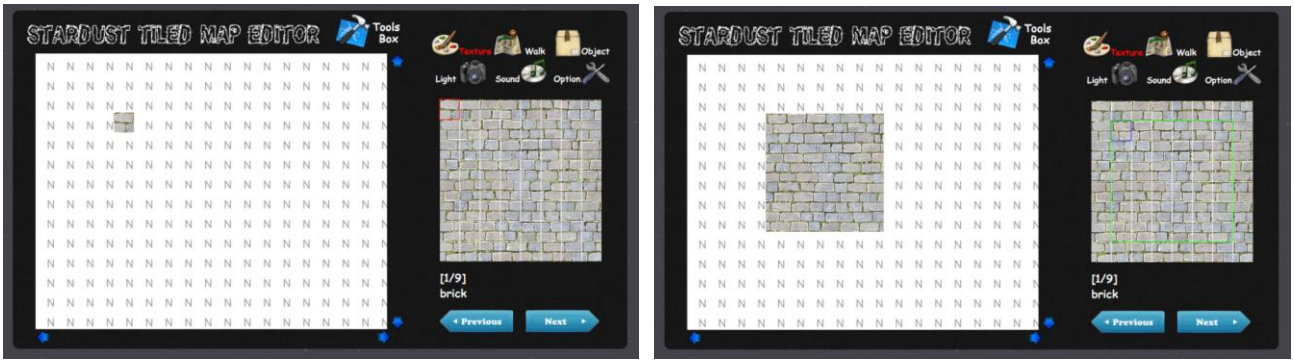
地圖預覽區的程式碼收錄在 build\src\build\map\preview_box.js 中。在 Material Box 建構子當中依照不同功能性質分成了四個部分，分別為：

物件性質	建構子	主要功能
MaterialBox.prototype.	OnCreate	繪製使用介面，並且導入其它編輯功能。
	OnTiledControl	地圖瓦磚管理，賦予不同座標點不同參數。
	GetToolsBox	物件工具箱，包含移動、縮放、排序和刪除。
	ToolsBoxListener	物件工具箱的點擊事件監聽。

表三、PreviewBox 的函數建構子。

（三）地圖紋理（Texture）

在地圖紋理的部分，提供了使用者在素材選取區選取不同素材、單張拼貼、多重拼貼等功能。



圖十與圖十一、設置拼塊素材。

1. 使用者介面（User interface in texture）

在素材預覽區中，在網頁開啟同時會先行全部載入原始圖檔，以利後續瀏覽速度加快。於 this.box.list.texture 將會顯示當前的地圖素材，預設值是 32*32pixel 正方形，共有 0~63 的索引點，利用 CreateJS 的 sourceRect 函數將其切割成矩形，因此這 64 個圖層彼此是獨立的方式拼貼而成。this.box.list.marked 將當前選取的索引值做高亮提示，單一選取時將會以紅色邊框、多重選取對任一處左鍵連續點擊兩次後會以藍色邊框提示，在點任一處後將其範圍構成矩形以

綠色邊框提示。this.box.list.page 管理素材的上一頁與下一頁。

build\src\build\map\material_box.js

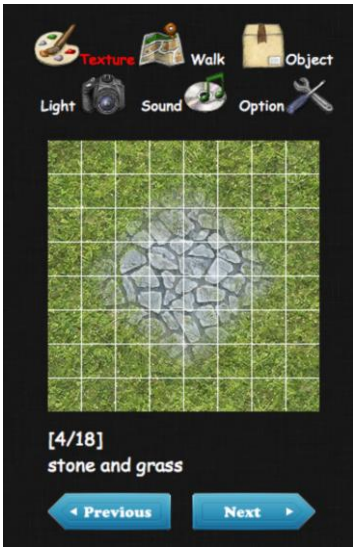
MaterialBox.prototype.OnTexture = function() {
1. 移除上一素材區 list Container 子元素，並將狀態轉為「OnTexture」。
2. 增加素材區（Container: texture）。
3. 此素材之上一頁、下一頁管理（Container: page）。
4. 選取範圍之提示邊框（Container: marked）。
5. 將以上宣告物件正式放置於素材區 list Container 中作為子元素。
6. 首次呼叫 Refresh()載入時刷新 Canvas。
}

圖十二、material_box.js 節錄建構子 OnTexture 執行步驟。

在地圖預覽區中，會根據繼承的 material 選取的地圖編號、選擇範圍之值，以進行紋理素材的拼貼。更動 OnTiledControl 當中的屬性，會呼叫 TotalRefresh 函數進行 Canvas layout 的瓦磚素材的重新繪圖。

2. 紋理資訊的資料結構（Data Structure in texture）

在 build\pic\map\texture 當中可以依照使用者自由增加想擴充的地圖素材，同一目錄下的 adding.js 文件當中將可以進行匯入圖檔管理，採取 JSON 格式做儲存。其中的 key 分別為：file 圖片檔名、decs 圖片描述，兩項屬性為單位的陣列方式儲存。



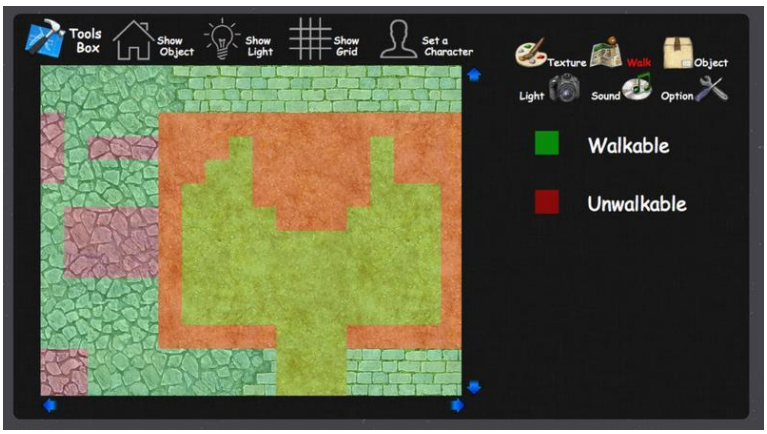
build\pic\map\texture\adding.js
var texture_adding = [{ file : "1.png", decs : "brick" }, …];

圖十三及圖十四、能夠自由切換不同素材，以及擴充地圖素材的管理文件。

地圖上的瓦磚為呈現此功能，將會需要記錄對應的圖片編號（見圖十三的 4/18）、圖片索引格數（見圖十三被劃分了 64 格，分別以索引值 0~63 表示），變數為每單位 tiled 的地圖編號「m」及該地圖索引值「i」。

（四）行走設定（Walkable）

在 OnTiledControl 每一格的瓦磚增加了是否可行走的屬性，在地圖預覽區的呈現部分將會以綠色遮罩表示可行走，紅色遮罩表示不可行走。



圖十五、地圖的瓦磚行走設定情形。

1. 使用者界面（User interface in walkable）

在素材預覽區中，在網頁開啟同時會先行全部載入原始圖檔，以利後續瀏覽速度加快。此段落補充說明與 **3-1 節**不同處，於 this.box.list.walkable 有兩個物件，分別控制「可行走」、「不可行走」的屬性，在點擊同時將會將顯示 this.box.list.marked 的提示邊框。

build\src\build\map\material_box.js

MaterialBox.prototype.OnWalkable = function() {
1. 移除上一素材區 list Container 子元素，並將狀態轉為「OnWalkable」。
2. 增加素材區（Container: walkable）。
3. 此素材之上一頁、下一頁管理（Container: page）。
4. 可行走、不可行走的文字提示設置（text1, text2）。
5. 選取之提示邊框（Container: marked）。
6. 將以上宣告物件正式放置於素材區 list Container 中作為子元素。
}

圖十六、material_box.js 節錄建構子 OnWalkable 執行步驟。

在地圖預覽區中，會根據繼承的 material 選取的行走與否之值。更動 OnTiledControl 當中的屬性，接著呼叫 TotalRefresh 函數進行 Canvas layout 更改行走提示的綠色及紅色遮罩之重新繪圖。

2. 行走資訊的資料結構（Data Structure in walkable）

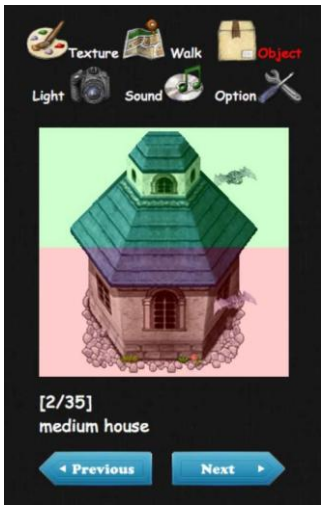
地圖上的瓦磚為呈現此功能，OnTiledControl 會需要記錄每一瓦磚的行走設定，將以 0 表示不可行走、1 表示可以行走，變數為每單位 tiled 的「w」。

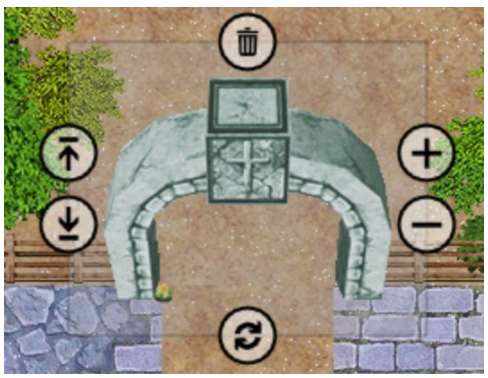
物件	變數縮寫及全寫		變數意義解釋
tileData	m	Number of map	紋理編號，對應 adding.js 紋理管理文件。
	i	Index in this map	於該地圖紋理素材的索引值。
	w	Walkable	座標允許行走之識別值。

表四、tileData 的資料結構整理。

（五）建築物件（Object）

建築物件可以任意的擺設於地圖範圍內的任意位置，並可對該物件進行「翻轉、縮放、物件間前後關係、捨棄」等功能。而建築物件提供了參考基準線，建築物件被劃分為綠色部分與紅色部分，綠色部分表示若有角色級別的圖層物件將會被建築物件覆蓋，紅色部分表示建築物件會被角色物件給覆蓋，藉此達到 2.5D 的畫面效果。





圖十七與圖十八、地圖物件的預覽情況及實際編輯情況。

1. 使用者界面（User interface in object）

在素材預覽區中，在網頁開啟同時會先行全部載入原始圖檔，以利後續瀏覽速度加快。於 this.box.list.objects 當中有物件圖片 pic、綠色遮罩 cover.coverer、紅色遮罩 cover.covered 以及覆蓋提示的兩個行動物件（見圖十七當中的兩隻飛行蝙蝠）。

build\src\build\map\material_box.js

```
MaterialBox.prototype.OnObject = function() {
    1. 移除上一素材區 list Container 子元素，並將狀態轉為「OnObject」。
    2. 增加素材區（Container: objects）。
    3. 此素材之上一頁、下一頁管理（Container: page）。
    4. 將以上宣告物件正式放置於素材區 list Container 中作為子元素。
}
```

圖十九、material_box.js 節錄建構子 OnObject 執行步驟。

2. 建築物件資訊的資料結構（Data Structure in object）

在 build/pic/map/object 當中可以依照使用者自由增加想擴充的建築素材，同一目錄下的 adding.js 檔案當中將可以進行匯入圖檔管理，採取 JSON 格式做儲存。其中的 key 分別為：file 圖片檔名、decs 圖片描述、divi 圖片百分比基準線，三項屬性為單位的陣列方式儲存。

build\pic\map\object\adding.js

```
var object_adding = [
    { file : "1.png",
      decs : "small house",
      divi : 0.50
    },
    ...
];
```

圖二十、圖片擴充建築素材的管理文件。

地圖上為附加建築物件於地圖上的功能，將會記錄圖片編號（見圖 O 的 2/35）、落點坐標

（為防止 js 原生浮點數低精准度問題衍生，輸出時會將其四捨五入）、物件大小與翻轉（1 代表原始大小、0.5 代表 1/2 大小、-1 代表水平翻轉，諸此類推）、物件間順序（依特殊呈現情況時仍可設定，但後續遊戲框架將會將所有物件依照 y 軸坐標作排序）。

物件	變數縮寫及全寫		變數意義解釋
objectData	n	Number of object	物件編號，對應 adding.js 物件管理文件。
	rx	Real x coordinate	網頁瀏覽器中真實的 x 座標。
	ry	Real y coordinate	網頁瀏覽器中真實的 y 座標。
	sx	Scale of horizontal	水平縮放倍率。
	sy	Scale of vertical	鉛直縮放倍率。

表五、objectData 的資料結構整理。

3. 建築物件工具箱（Tool box）

於 preview_box.js 中的 GetToolsBox 提供了地圖預覽區上的物件編輯功能。任一個物件被放置預覽區的同時將會呼叫 preview 的建構子 GetToolsBox，讓該物件擁有可被編輯、修改的功能。其提供的功能如下：

建構子	函數	主要功能解釋
GetToolsBox	cancel	刪除物件。
	flip	水平翻轉。
	up	排序提升一層。
	down	排序降低一層。
	zoom_in	整體縮放上升。
	zoom_out	整體縮放下降。

表六、Tools box 的功能函數整理。

當左鍵點擊物件後將可展開工具箱介面，移出工具箱範圍將會自動隱藏工具箱。

（六）燈光特效（Light）

利用軌跡公式運算物件的流線型軌跡達到動畫特效，此處提供了幾個範例演示。將繼承建築物件除了遠近關係以外之可做功能，意即燈光特效物件的優先權高於建築物件，將會置於建築物件之上方。

圖二十一、燈光特效的使用實例。

1. 使用者界面（User interface in light）

在開啟網頁時，會先行導入文件 light_effect.js，這份文件當中寫入了由小型的圖形粒子構成的微型動畫，利用簡易的座標改變達到類似燈光特效的效果。

build\src\build\map\material_box.js

0. 先行導入文件 light_effect.js。
MaterialBox.prototype.OnLight= function() {
1. 移除上一素材區 list Container 子元素，並將狀態轉為「OnLight」。
2. 增加素材區（Container: light）。
3. 此素材之上一頁、下一頁管理（Container: page）。
4. 將以上宣告物件正式放置於素材區 list Container 中作為子元素。
}

圖二十二、material_box.js 節錄建構子 OnLight 執行步驟。

2. 軌跡公式（Animation function）

在 light_effect.js 當中，提供了幾個基本例子作為拋磚引玉讓未來其他開發者能夠自行撰寫相關程式。使用 CreateJS 當中的 Tween 套件可以做出繪圖元素的小型動畫，以下附上圖 O 的特效範例程式碼。

build\src\build\map\light_effect.js
function LightEffect(effect_num) { var effectContainer = new createjs.Container() ; effectContainer.name = effect_num ;

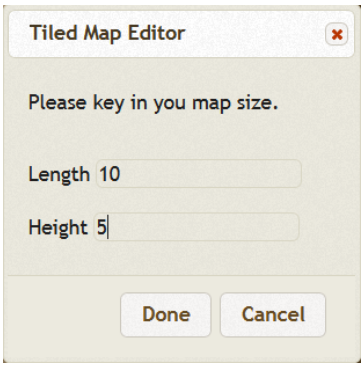
```
    if ( effect_num == 1 ) {
        effectContainer.regX = 150, effectContainer.regY = 100 ;
        effectContainer.setBounds( 0, 0, 300, 200 ) ;
        for ( k = 0 ; k < 30 ; k ++ ) {
            var grain = new createjs.Shape() ;
            grain.graphics.f( "#FFFFFF" ).drawPolyStar( 0, 0, 10, 10, 0.6, -90 ) ;
            effectContainer.addChild( grain ) ;
            grain.x = 150, grain.y = 0 ;
            grain.alpha = 0 ;
            createjs.Tween.get( grain, { loop: true } )
            .wait( RandomTime( 200, 2000 ) )
            .to( { alpha: 0.5 }, 0 )
            .to( { x: RandomRange( 300 ), y: 200, alpha: 0, rotation: 360 }, 2000 ) ;
        } // for
    } // if
    …
}
```

圖二十三、light_effect.js 節錄，圖二十一特效之程式碼。

（七）地圖輸出、輸入（Export & import）

1. 新開地圖（New map）

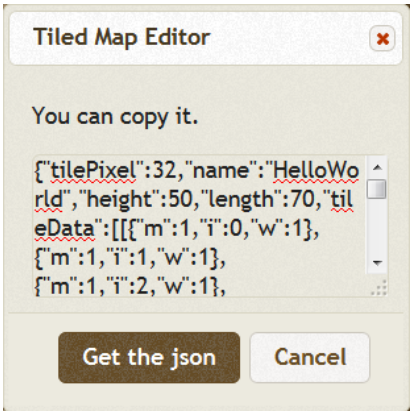
能夠自由選擇地圖的長與寬，超出可視範圍可利用地圖預覽區右側及下側的游標進行畫面移動。



圖二十四、地圖編輯器的新開地圖演示。

2. 輸出地圖（Export a map）

輸出的 JSON 可自行貼上至地圖管理文件即可新增該地圖。



圖二十五、地圖編輯器的輸出演示。

3. 輸入地圖（Import a map）

將地圖 JSON 檔貼上至對話方框中將可二度編輯該地圖。

六、主要遊戲框架

（一）遊戲架構規劃（Structure plan）

遊戲客戶端程式放置於 client 資料夾中，其中之 game.html 為啟動遊戲的頁面。其他子資料夾因功能分類後如下：

資料夾路徑	資料夾內檔案類型
src\RPG\	遊戲主程式。
character\body\	角色物件的軀幹模組素材。
character\hair\	角色物件的髮型模組素材。
map\map_texture\object\	地圖素材之建築物件。
map\map_texture\texture\	地圖素材之地圖紋理。
npc\npc_src\	NPC 物件之腳本。
npc\npc_texture\	NPC 物件之模組素材。
pic\monster\	怪物物件之模組素材。
sound\BGM\	背景音樂。
sound\monster\	怪物行動之音效。
sound\skill\	技能施放之音效。

表七、client 的各子資料夾分類。

於 src\RPG\路徑中的遊戲主程式，同時在 game.html 開啟時匯入，各個檔案負責之部分如下：

src\RPG\目錄中檔案	各檔案主要負責項目
global_function.js	全域函數，在各個檔案中會被重複呼叫的數學函數。
map.js	地圖繪製以及行走控制函數。
sprite_choose.js	角色級模組的播放動畫。
character.js	玩家控制角色。
npc.js	非玩家控制角色。
framework.js	遊戲輔助視窗。
server_connection.js	socket 連線插件。

表八、client 的各子資料夾分類。

後續將會重複出現一些制式名詞，在此先行定義兩物件名詞。

地圖級物件	包含建築物件、燈光特效物件
角色級物件	包含玩家角色、其他玩家角色、非玩家操控角色

（二）精靈動畫（Sprite）

Sprite 是一微型動畫的技術，利用僅一張圖檔依照不同座標切割成不同圖層（frame），再進行連續播放不同圖層間達到酷似動畫的技術，能夠有效減少圖檔間的檔案容量大小。在 sprite_choose.js 中記錄了角色級物件的 Sprite 的動畫圖層播放順序。

1. 角色級玩家物件 Sprite（Sprite of player characters）

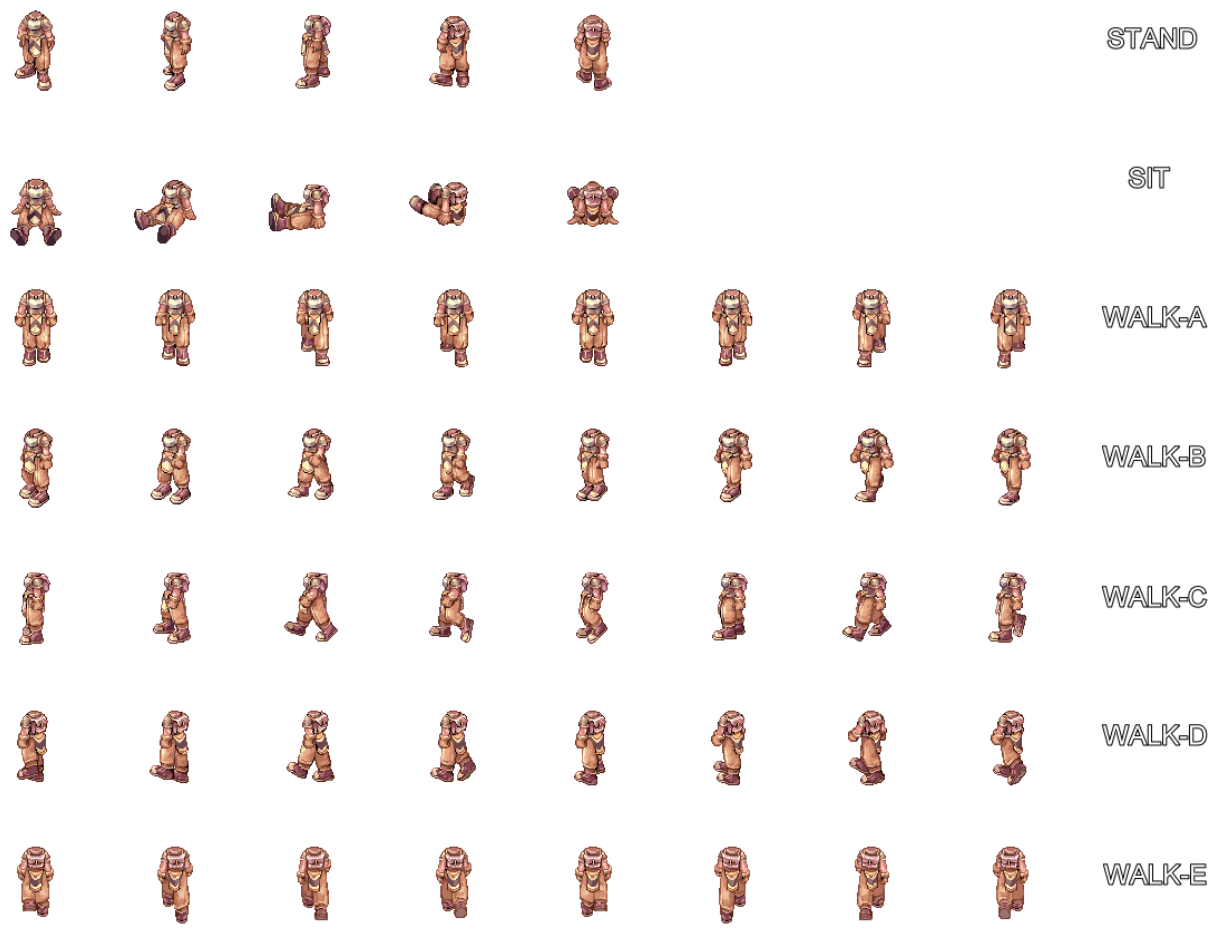
利用 CreateJS 的 SpriteSheet 函數，能夠快速生成 sprite 動畫。玩家角色 animations 之狀態能夠針對不同情況進行轉換，其中包含不同方向的站姿、坐姿、行走等動作。角色被劃分為八個面向，而圖檔中儲存五個面向，剩餘的三個面向分別為右前方、右方及右後方，將由對應之左前方、左方及左後方鏡射生成。玩家角色 sprite 區分了軀幹以及髮型，軀幹的每個圖層為 125*125 pixel、髮型的每個圖層為 75*75 pixel，玩家角色將可以擁有不同的搭配組合產生多種的角色紙娃娃。

client\src\sprite_choose.js

```
SettingSprite = function( type, name ) {
    var spriteSheet ;
    if ( type.target == "character" ) {
        if ( type.part == "body" ) {
            if ( name == "novice" )
                spriteSheet = new createjs.SpriteSheet( {
                    "images": ["body/novice.png"],
                    "frames": { "width": 125, "height": 125, "regX": 0, "regY": 0, "count": 62 },
                    "animations": {
                        "stand_A": { "frames": [0], "speed": 1 },
                        "stand_B": { "frames": [1], "speed": 1 },
                        "stand_C": { "frames": [2], "speed": 1 },
                        "stand_D": { "frames": [3], "speed": 1 },
                        "stand_E": { "frames": [4], "speed": 1 },
                        "sit_A": { "frames": [9], "speed": 1 },
```

```
                                "sit_B": { "frames": [10], "speed": 1 },
                                "sit_C": { "frames": [11], "speed": 1 },
                                "sit_D": { "frames": [12], "speed": 1 },
                                "sit_E": { "frames": [13], "speed": 1 },
                                "walk_A": { "frames": [18,19,20,21,22,23,24,25], "speed": 0.13 },
                                "walk_B": { "frames": [27,28,29,30,31,32,33,34], "speed": 0.13 },
                                "walk_C": { "frames": [36,37,38,39,40,41,42,43], "speed": 0.13 },
                                "walk_D": { "frames": [45,46,47,48,49,50,51,52], "speed": 0.13 },
                                "walk_E": { "frames": [54,55,56,57,58,59,60,61], "speed": 0.13 }
                            }
                        } ) ;
            } // if
            ...
        } // if
}
```

圖二十六、sprite_choose.js 節錄。角色級物件 novice 的軀幹 sprite 模組圖層播放順序。



圖二十七、角色級物件 novice 的軀幹 sprite 模組圖檔。

client\src\sprite_choose.js

```
if ( type.target == "character" ) {
        if ( type.part == "hair" ) {
                if ( name == "style1_white" )
                        spriteSheet = new createjs.SpriteSheet( {
                                "images": ["hair/white.png"],
                                "frames": { "width": 75, "height": 75, "regX": 0, "regY": 0, "count": 15 },
                                "animations": {
                                        "stable_A": { "frames": [0], "speed": 1 },
                                        "stable_B": { "frames": [1], "speed": 1 },
                                        "stable_C": { "frames": [2], "speed": 1 },
                                        "stable_D": { "frames": [3], "speed": 1 },
                                        "stable_E": { "frames": [4], "speed": 1 }
```

```
                    }
                });
    } // if
} // if
```

圖二十八、sprite_choose.js 節錄。角色級物件 white 的髮型 sprite 模組圖層播放順序。

圖二十九、角色級物件 white 的髮型 sprite 模組圖檔。

2. 角色級非玩家物件 Sprite（Sprite of non-player characters）

非玩家物件基礎為單一面向或前後面向，只要遵照圖層索引值的數量作控制及取用即可。非玩家物件的 sprite 每圖層為 125*125 pixel。與玩家物件不同處還有一點，非玩家物件在圖檔資料夾新增了角色的象徵圖可以採用，只要遵照 sprite 檔案名稱後加上「_l」後，在設計 NPC 腳本時可以開啟對話時顯示象徵圖，屆時將會自動讀取該圖片。

client\src\sprite_choose.js

```
else if ( type.target == "npc" ) {
    if ( name == "sage" )
        spriteSheet = new createjs.SpriteSheet( {
            "images": ["npc_texture/sage.png"],
            "frames": { "width": 125, "height": 125, "regX": 0, "regY": 0, "count": 36 },
            "animations": {
                "front": { "frames": [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17], "speed": 0.1 }
            }
        } );
}
```

圖三十、sprite_choose.js 節錄。角色級物件 sage 的 NPC sprite 模組圖層播放順序。

圖三十一、角色級物件 sage 的 NPC sprite 模組圖檔與角色象徵圖。

3. 角色級怪物物件 Sprite（Sprite of monsters）

怪物物件亦是非玩家物件一環，主要是為了分類上才有所區別，非玩家物件多有與玩家對話上的互動，怪物物件則無。除此之外兩者的模組亦能夠相互共用，其 sprite 每圖層亦為 125*125 pixel 與 NPC sprite 相同。

client\src\sprite_choose.js

```
else if ( type.target == "monster" ) {
    if ( name == "farmiliar" )
        spriteSheet = new createjs.SpriteSheet( {
            "images": ["monster/farmiliar.png"],
            "frames": { "width": 125, "height": 125, "regX": 0, "regY": 0, "count": 36 },
            "animations": {
                "front": { "frames": [0,1,2,3,4,5,6,7], "speed": 0.3 },
                "walk": { "frames": [0,1,2,3,4,5,6,7], "speed": 0.3 }
            }
        } );
}
```

圖三十二、sprite_choose.js 節錄。角色級物件 farmiliar 的怪物 sprite 模組圖層播放順序。

圖三十三、角色級物件 farmiliar 的怪物 sprite 模組圖檔。

（三）玩家角色（Player character）

玩家角色分為玩家本身與其他玩家，兩者的差異僅為控制與否。在本節當中將會介紹玩家角色物件 Character 的生成方式，及經過模組化的行動指令。

圖三十四、分別執行不同命令的 Character 物件。

1. 物件 Character 的結構（Structure of Character）

物件性質	建構子	主要功能
Character.prototype.	OnCreate	建立角色物件至地圖中，並生成其 sprite。
	OnActive	玩家本身角色增加滑鼠與鍵盤事件監聽。
	OnMove	直接移動角色至指定座標。
	OnWalk	角色會行走至指定座標。
	OnDirection	改變角色 sprite 播放的 animation。
	OnTalk	角色頭頂上出現對話泡泡框。
	OffTalk	隱藏對話泡泡框。
	OnPlaySound	角色播放指定動作之音訊。

LifeBar	建立角色顯示生命血條於腳下。
OnLifeModify	修改生命值。
OnEffect	角色播放特效。
OnSkill	角色施放技能。
TimeSleep	自身計時延遲函數。

表九、Character 的函數建構子。

2. 物件事件監聽（OnActive）

若要控制角色物件隨著游標點擊之處移動，將會需要監聽其滑鼠點擊事件。於 Canvas 最高層級物件 stage 中追加 stagemousedown 滑鼠點擊事件，利用傳送至角色物件的 MapControl. MapControlPointer 經過計算後取得預移動之座標點並呼叫 OnWalk 函數進行行走動作。而計算方式為以下步驟所述：

(1) 取得 Canvas 中實際 x、 y 座標值。以當前畫面可見的左上方第一瓦磚座標為(1, 1)，將取得之實際座標轉為瓦磚座標 trimedGrid。

(2) 取得當前畫面可見的左上方第一瓦磚正確之座標，並以修正方式補加在 trimedGrid 的 x、y 的瓦磚座標上，此時 trimedGrid 即為正確的目標瓦磚座標。

client\src\RPG\character.js

```
stage.on( "stagemousedown", function( evt ) {
    if ( that.MapControlPointer.nowEventTrigger == null ) {
        var trimedGrid = that.MapControlPointer.GetGrid( { x: evt.stageX, y: evt.stageY }, "real" ) ;
        trimedGrid.x += that.MapControlPointer.trim.x,
        trimedGrid.y += that.MapControlPointer.trim.y ;
        that.OnWalk( trimedGrid ) ;
    } // if
    else if ( that.MapControlPointer.nowEventTrigger != "TriggerNow" )
        that.MapControlPointer.nowEventTrigger.OnTrigger() ;
}) ;
```

圖三十五、character.js 節錄。Character 的點擊事件判斷其目標座標。

3. 物件移動（OnMove）

在執行此指令時，系統會先行判斷角色物件所處地圖是否可允許在該點上行走，確認可行

走後將修改角色物件的瓦磚 x、y 座標，一併移動角色物件的 sprite 所處畫面位置，最後進行畫面上的全體物件排序。

client\src\RPG\character.js

```
if ( this.MapControlPointer.MapMove( { x: endGrid.x, y: endGrid.y }, 0 ) ) {
    this.container.grid_x = endGrid.x, this.container.grid_y = endGrid.y ;
    var realGrid = this.MapControlPointer.GetGrid( { x: endGrid.x, y: endGrid.y }, "virtual" ) ;
    this.container.x = realGrid.x + this.container.regX,
    this.container.y = realGrid.y + this.container.regY * 0.3 ;
    this.resortingOrder() ;
} // if
```

圖三十六、character.js 節錄。Character 的直接移動至指定座標。

4. 物件行走（OnWalk）

傳入預前往的瓦磚座標，將會把當前座標 startGrid 以及目標座標 endGrid 傳至 MapControl 的 MapControlPointer.AStarAlgorithm 進行最短路徑計算，計算結果以陣列方式存入 pathfinding。在 pathfinding 陣列資料為格格相連的連續八方位之值（0~7 由上方作順時鐘旋轉），以遞迴方式連續將 pathfinding 所有路徑取出同時進行物件移動，移動期間會依照面向方向不同，傳入對應之值至 OnDirection 函數改變顯示的 sprite。

client\src\RPG\character.js

```
var pathfinding = that.MapControlPointer.AStarAlgorithm( startGrid, endGrid ) ;
Recursive( 0, pathfinding.length ) ;
function Recursive( nowPath, totalPath ) {
    1. 依照不同行走方向改變座標差值 distanceX 與 distanceY。
    2. 角色速度補正。
    3. 檢查是否繼續行走，若行走則改變 sprite 為行動 animation。
    4. 結束行走時，已結束面向改變 sprite 為站立 animation。
}
```

圖三十七、character.js 節錄。Character 的行走至指定座標是藉由 AStarAlgorithm 函數計算。

5. 物件 sprite 動畫（OnDirection）

玩家角色級物件的 OnDirection 會需要控制軀幹及髮型的不同方位，因此傳入的參數不僅有方位(direction)還會有位置類型(type)，並依照當前會需要播放的動畫類型 currentAnimation 挑選正確的 sprite 圖層。

client\src\RPG\character.js

```
if ( type.part == "body" ) {
	this.sprite.body.scaleX = ( direction != -1 ) ? ( ( direction > 0 && direction <= 4 ) ? -1 : 1 ) : direction ;
	// If not the same animation must change.
	if ( this.sprite.body.currentAnimation != type.mode )
		this.sprite.body.gotoAndPlay( type.mode ) ;
} // if
else if ( type.part == "hair" ) {
	this.sprite.hair.scaleX = ( direction != -1 ) ? ( ( direction > 0 && direction <= 4 ) ? -1 : 1 ) : direction ;
	// If not the same animation must change.
	if ( this.sprite.hair.currentAnimation != type.mode )
		this.sprite.hair.gotoAndPlay( type.mode ) ;
} // else if
this.container.direction = ( direction != -1 ) ? direction : this.container.direction ;
```

圖三十八、character.js 節錄。Character 改變當前播放動畫圖層。

6. 物件對話泡泡框（OnTalk & OffTalk）

對話泡泡框會依照輸入的對話內容，判定全行及半形字元數量後進行長度加權，繪製合適的矩形長度，此處採用的顯示字型為等寬型字體，以避免不同半形字元寬度不一。在觸發 OnTalk 時會一併呼叫 OffTalk 倒數計時以利消除對話框。連續進行對話的處理方式為，將會累加對話框保留時間，並修正對話內容後顯示。

client\src\RPG\character.js

```
this.OffTalk( 'now' ) ;
var chat_len = 20 + ( halfFullCheck( "half", text ) * 1.07 + halfFullCheck( "full", text ) * 1.71 ) * 10 ;
this.talk.bg.alpha = 0.65 ;
this.talk.bg.graphics.f( "#000" ).r( 0, 0, chat_len, 25 ) ;
this.talk.wd.text = text ;
```

```
this.talk.wd.x = 10, this.talk.wd.y = 6 ;
this.talk.wd.alpha = 0.9 ;
this.talk.x = 67 - chat_len / 2, this.talk.y = -15 ;
var that = this ;
createjs.Tween.get( that.container, { loop: false } ).call( function() { that.OffTalk( 'fade' ) ; } ) ;
```

圖三十九、character.js 節錄。Character 的 OnTalk。

client\src\RPG\character.js

```
if ( type == 'now' ) {
	this.talk.wd.text = "" ;
	this.talk.bg.graphics.c() ;
} // if
else if ( type == 'fade' ) {
	var that = this ;
	this.talk.fadetime += 3000 ;
	this.TimeSleep( function() {
		that.talk.wd.text = "" ;
		that.talk.bg.graphics.c() ;
	} ) ;
} // else if
```

圖四十、character.js 節錄。Character 的 OffTalk。

（四）非玩家角色（Non-Player character）

非玩家角色繼承了玩家角色物件的所有建構子，在此段落將不再額外說明各函數之功能。

1. 腳本撰寫（Scripting）

NPC 能夠自由與用自身的建構子函數，便能在設計 NPC 時依照需求調用不同的函數來達到有劇情脈絡的結果。以下的範例腳本為玩家觸發對話時，NPC 會出現對話框以及顯示角色象徵圖。

client\npc\adding_npc.js

```
var NPC_Function_sage = function( that ) {
	that.OnTalk( that.container.name + ": God will bless you and me." ) ;
	that.OnCutin( "npc_texture/sage_l.png", 1 ) ;
}
```

圖四十一、一支會同時出現對話框與象徵圖的 NPC 腳本。

2. 同步與非同步問題（Asynchronous problem）

觀察上一小節程式碼可發現兩道指令會「同時觸發」，此為原生 JavaScript 語言特性會發生的同步問題，由於在設計腳本時會需要有停頓的過程，必須引入非同步的概念，在此使用了 Async.js 此款套件解決。每經過 checkTime 毫秒將會檢查是否有點選 _dialogNext 的下一步按鈕，直到玩家點擊下一步或是離線才會進行下一步或是中止對話。

client\npc\adding_npc.js

```
var NPC_Function_sage = function( that ) {
	var checkTime = 100 ;
	async.series([
		function( callback ) {
			that._dialogNext = false ;
			setTimeout( function() {
				that.OnDialog( { first: "Hello." } ) ;
				callback( null, 'one' ) ;
			}, 0 ) ;
		},
		function( callback ) {
			Loop( checkTime ) ;
			function Loop( checkTime ) {
				setTimeout( function() {
					if ( that._dialogNext ) {
						that._dialogNext = false ;
						that.OnDialog( { second: "Nice to meet you." } ) ;
					} // if
					else
						Loop( checkTime ) ;
				}, checkTime ) ;
			} // Loop()
			callback( null, 'two' ) ;
```

```
        },
        ...
}
```

圖四十二、加入了非同步概念的 NPC 腳本。

3. 新增 NPC（Add NPC）

於路徑 client\npc 中的文件 npc_adding.js 當中記錄了要被套用的 NPC。文件中的 func 為 NPC 的函數名稱，map 為所在地圖，grid 為瓦磚座標，name 為顯示名稱，sprite 為套用的 sprite 模組，direct 為預設面向。

client\npc\adding_npc.js

```
var npc_adding = [
    {   func : NPC_Function_sage,
        map      : "village",
        grid  : { x: 33, y: 8 },
        name     : "Sage",
        sprite   : { type: "npc", name: "sage" },
        direct   : 7
    },
    ...
};
```

圖四十三、NPC 的管理文件。

七、結論與未來方向

（一）總結

能有這一次難忘的專題經驗實屬開心，在過程中自我挑戰嘗試了未曾使用過的語言、套件以及技術。在設想不同的遊戲環境可能會遇到的問題，2.5D 的電腦圖學問題、A-Star 的最短路徑演算法也尋求不少老師、學長及同學們的幫助，深感欣慰。澤浩於文末深深感謝指導老師吳宜鴻老師適時給予幫助及建言，總能令我如醍醐灌頂般。謝謝另外兩位不吝指導的蘇志文老師與朱守禮老師。謝謝已畢業的張祐翔、林緯謦、林耿義三位學長在使用套件與熟悉語言上給予相當大的輔助。以及同屆同學郭大維、鄭亦茵亦提供了相當良好的專題建議。最後特別感謝台灣大學網媒所王瀚宇學長的提攜。

這個專題有了一個美好的開始，卻有些虎頭蛇尾般的結束，在進行專題研究的這兩年發生了很多不在自己預料內的事情。謝謝吳宜鴻老師的多番包容，不論是多次轉換專題方向還是時而未達進度標準。過去的已無法改變，警惕自己未來要更看重每件交付在自己身上的任務！

（二）未來方向

1. 支援環境音效（Sound）

針對特定坐標的方圓特定半徑內提供環境音效的播放，例如接近溪流會聽見水聲，靠近樹林會有蟲鳴鳥叫。

2. 非玩家角色編輯器（NPC Editor）

將已模組化的程式已圖形化的方式呈現，利用不同的拼塊組合成一 NPC 的完整腳本。

八、參考資料

[1]：rAthena 官方網站，
http://rathena.org/board/。
[2]：RPG Maker 官方網站，
http://www.rpgmakerweb.com/。
[3]：2.5D（two and a half dimensional），
http://en.wikipedia.org/wiki/2.5D。
[4]：RPG Maker 的地圖編輯器畫面，
http://blog.rpgmakerweb.com/tutorials/make-your-own-game-part-1/。
[5]：Scratch 官方網站，
http://scratch.mit.edu/。
[6]：Scratch 的操作畫面，
http://gpio.kaltpost.de/?page_id=1558。

所使用的相關套件與程式

1. 網頁後端伺服器：
   Node.js，http://nodejs.org/
2. 前端 Canvas 套件：
   CreateJS，http://www.createjs.com/
3. 異步處理套件：
   Async.js，https://github.com/caolan/async
4. 前端 DOM 操作套件：
   jQuery & jQuery UI，http://jquery.com/
5. 連線 Web Socket 套件：
   Socket.IO，http://socket.io/
6. 資料庫 Database：
   MySQL & MySQL Workbench，http://dev.mysql.com