國立台灣科技大學
資訊工程系

碩士學位論文

以遊玩特徵為導向的程序化內容生成方法

Game Design Goal Oriented Approach for Procedural Content Generation

研 究 生：王澤浩
學　　號：M10415096

指導教授：戴文凱博士

中華民國一百零六年七月二十七日

中文摘要

中文摘要，編輯中。

ABSTRACT

Ab. Under construction.

誌　　謝

誌謝，頁面保留。

目　　錄

圖　　目　　錄

表　　目　　錄

第 1 章　緒論

程序化內容自動生成 (Procedural Content Generation) 在過去就廣泛被應用於遊戲設計領域，其主要目的為增加遊戲內容的隨機性與多樣性。在本文中，我們針對遊戲過程中的遊玩特徵 (gameplay patterns) 進行抽象化，使用程序化生成技術產生帶有意義遊戲關卡內容，藉此消彌或降低因隨機性所產生的不穩定要素，以改善並豐富遊戲體驗。

我們將遊戲關卡的構成劃分為任務 (Missions) 與空間 (Space) 兩種結構後，空間會依照任務結構進行有意義的轉換，接著依照遊玩特徵定義基因演算法 (Genetic Algorithms) 的演化依據。讓玩家在進行遊戲時能夠遵循關卡設計師的劇情脈絡外，亦能夠體驗到有意義且多樣化的遊戲關卡內容。

1.1　研究背景與動機

content here.

1.1.1　迷宮探索遊戲 (dungeon crawl) 類型介紹

content here.

迷宮探索遊戲的歷史

content here.

1.2 研究目的與研究問題

content here.

1.2.1 研究目的

content here.

1.2.2 研究問題

content here.

1.2.3 預計研究貢獻

content here.

1.3 本論文之章節結構

content here.

第 2 章　相關研究

content here.

2.1　含有敘事脈絡的關卡

Jenova Chen 在 [1] 提到，心理學領域的心流理論 (Flow) 應如何被運用到遊戲設計中，且藉由該理論來改善遊戲設計中的..... (編輯中)。

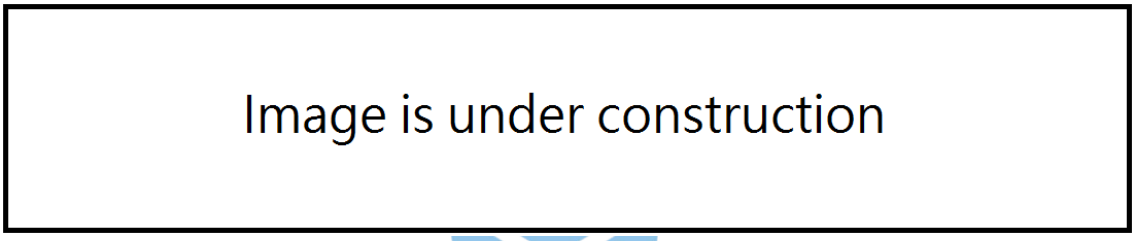

圖 2.1: 心流體驗應用於遊戲設計

2.2　關卡生成的方法 (Level Generation)

本節收錄了至今的關卡生成框架與方法。而參考的文獻主要分為兩種類型，分別為 2.2.1 小節的程序化生成任務內容與 2.2.2 小節的程序化遊戲物件擺放。同時收錄數款採取關卡生成技術之遊戲。

2.2.1　程序化生成任務內容

content here.

Mission/Space 框架

Joris Dormans 認為一個完整的關卡需要包含任務與空間二者 [2] [3] [4]；需要有一特定的空間佈局，及一系列需要於此空間中被執行的任務。關卡任務代表玩家需要按照任務流程，來依序挑戰才能夠完成該關卡；關卡的空間由其地理佈局所組成，或者由與地圖相似的節點網絡所構成。由於任務與空間之間的交錯混雜，導致關卡設計者最終採取簡單卻有效的策略，也就是讓任務與空間同構。雖然同構在設計上不是唯一的選擇，但對於某些遊戲是非常合適的，特別是一具有線性的關卡設計。而 Joris Dormans 亦提出了一種自動關卡設計的方法 [2] [4]，藉由產生一個任務，再利用這個任務去產生適合此任務的空間。舉例來說，關卡設計者透過生成任務的介面來建立任務圖 (mission graph)，玩家必須執行這些任務才能夠完成關卡，接下來將任務轉換為空間，並將任務依序安排至該空間圖 (space graph) 中。設計者接著在地圖添加更細節的內容，直到地圖充滿任務的要素並作為遊戲的關卡。

任務圖注重於任務與玩家的相互關係，表現出玩家距離通關的進度狀況。主要由兩種要件：節點和有向連結線所構成，其中節點再細分為任務、起點與終點；有向連結線再依照兩節點之間的執行先後關係，細分為薄弱條件、強烈條件與抑制。其中，強烈條件或抑制的關係，會導致某些節點無法執行。空間圖直接呈現了關卡的空間結構，且大多數的節點能夠直接表示出玩家目前所在位置。空間圖中的任何節點能透過顏色、字母來表示不同類型。主要亦由兩種要件：節點和連結線所構成。節點細分為場所、鎖和遊戲元素所構成；有向連結線細分為通道、閥、窗、解鎖與上鎖等。

改寫系統 (rewrite system) 由具有左側與右側的規則 (rules) 所組成，能夠將規則中指定的一符號集能夠被另一符號集所取代。改寫規則當中所使用的符號，便是在遊戲中經常會出現一些具有代表性的物件、要素或任務目標等，在字符表 (alphabet) 中定義以抽象化描述遊戲中的週期性結構 (recurrent construction)。改寫系統能夠套用在構成任務的圖形語法 (graph grammars) 及構成空間的形狀語法

(shape grammars)，二者能夠獨立生成出結果，不過建議能夠將改寫系統套用在任務圖上，使其能夠產生出空間圖。本文提及之任務圖和空間圖是經過改良後的版本，定義其規則時會有些微上的不同，但更能夠體現出遊戲的關卡結構。

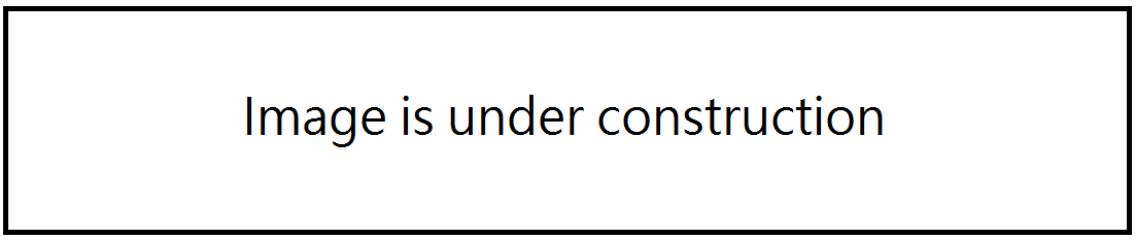

圖 2.2: Mission/Space 框架之主要結構

Dungeon Crawl

content here.

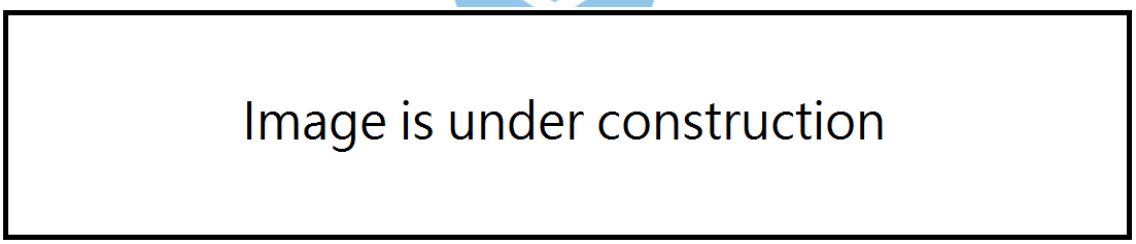

圖 2.3: Unexplored 的關卡設計

2.2.2 程序化遊戲物件擺放

content here.

Map Sketches 與 **Segments** 的演化

Antonios Liapis 開發了策略型遊戲的抽象化地圖生成工具 - Sentient Sketchbook [5]。在 Sentient Sketchbook 中，遊戲關卡設計師能夠以低分辯率、高階抽象的方式來

編輯地圖草圖 (map sketches)，構成地圖的瓦磚類型有資源磚、基地磚、不可通行磚與可通行磚等。典型的戰略型遊戲中，每位玩家都必須從隨機選擇的基地開始採集資源以建構戰鬥單位，並利用這些戰鬥單位摧毀敵方基地以完成遊戲。

當設計師編輯地圖時，該工具能夠測試地圖的可玩性 (意旨能夠正常進行遊戲) 且量化顯示，如果沒有足夠的基地、資源或可連通的路徑，那麼工具提供的遊玩特徵指標將會提示該地圖為不可遊玩的狀態。而這些遊玩特徵指標分別為資源安全性：距離基地僅一格以內的資源磚數量；安全區域：計算基地與敵方基地間的磚總數；探索性：利用洪水填充演算法，計算從基地至敵方基地時，可通行的磚總數。透過用戶當前編輯的地圖草圖，該工具利用基因演算法進行前述等指標，評估適應性函數 (fitness functions) 以解決約束最佳化 (constrained optimization) 等問題，來產生出更多意想不到的地圖輸出結果。

後續的研究中，Antonios Liapis 將基因演算法調整為兩階段演化，第一階段演化為地圖草圖演化，第二階段為地圖片段 (map segments) 演化 [6]。地圖片段的結構類似於地圖草稿，由 NxN 的瓦磚所構成，瓦磚的種類能夠像是空磚、牆、連接處、出口、怪物或寶箱等，其中連接處是為了讓地圖片段彼此能夠接合以填滿地圖。利用地圖草稿所轉化成的初始地圖片段可作為演化用的胚胎 (embryogeny)，於此階段定義的牆、連接處會呈顯穩定狀態，不隨著演化過程而改變，其餘瓦磚有機會由空磚突變為怪物、寶箱或牆，反之亦然。並探討不同的目標函數與胚胎，如何影響的圖片段的最佳解與外觀。

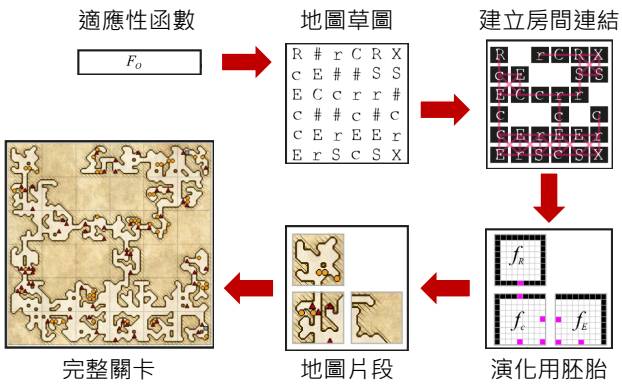

圖 2.4: Antonios Liaps 提出的兩階段式關卡演化

2.3 使用關卡生成技術的遊戲

content here.

遊戲 - **Unexplored**

Unexplored 是由 2.2.1 小節提及之框架作者 Joris Dormans 所製作，Joris Dormans 基於 Mission/Space 框架並提出改良的即時型戰鬥 roguelike 遊戲（若再細分可歸類於 roguelite 遊戲類型）。在市面上普遍的 roguelike 遊戲中，最常見的關卡生成策略即是視地下城玩家起點位置為樹之根，再以此根持續添加生成或預先設計的

Joris Dormans 稱改良後的方法為環狀地下城生成 (cyclic dungeon generation)，在每一層的地牢中...

2.3.1 遊戲 - Spelunky

content here.

2.3.2 遊戲 - The Binding of Isaac

content here.

第 3 章　研究方法

本論文中，我們仍保持 Joris Dormans [2] [4] 與 Antonios Liapis [5] 為求遊戲設計過程抽象化與高階化的訴求。將「Mission/Space 框架」與「Multi-segment 演化」兩種關卡生成方法結合並予以改良，保留了前者追求的遊戲進程之順序性，後者帶來穩定且多樣化的遊戲內容，希冀藉此提升整體遊戲體驗、相輔相成。

圖 3.1 為系統的整體流程圖。遊戲設計師能夠透過宏觀的觀點來構築遊戲體驗流，將遊玩特徵拆分成多項規則，利用生成語法及改寫系統生成出任務圖，圖 3.1-a。依照任務圖中對應的終端節點 (terminal nodes) 轉換為事先建構完成的房型空間，並得到尚未包含遊戲物件的遊戲地圖，圖 3.1-b 至 c。接下來，針對動作冒險遊戲我們提出了數項評估遊戲性的適應性函數，藉由基因演算法的演化流程，令各房間遵守適應性函數的限制，以得到符合訴求的最適解，圖 3.1-c 至 e。

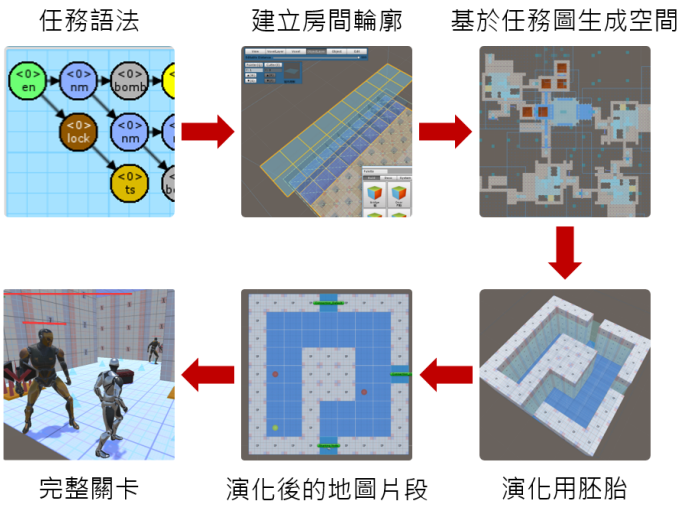

圖 3.1: 本論文提出系統之流程圖

3.1 任務語法

我們基於 Joris Dormans 提出之 Mission Grammars 概念，進行實作與改良出 Dungeon Generator 工具，這項工具佈署在 Unity Engine 上。遊戲設計師能夠藉由 Dungeon Generator 工具進行任務語法的建置，並執行改寫系統以輸出任務圖，進一步利用任務圖產出遊戲關卡空間。

在任務語法的設計階段，我們參考 2.3.2 小節所提及之遊戲 **The binding of Issac** 的關卡地圖，分析其遊戲進程結構，構想期望的任務圖並將觀察其遊玩特徵，接著拆分成任務語法規則使用改寫規則產生出近似結構的任務圖。

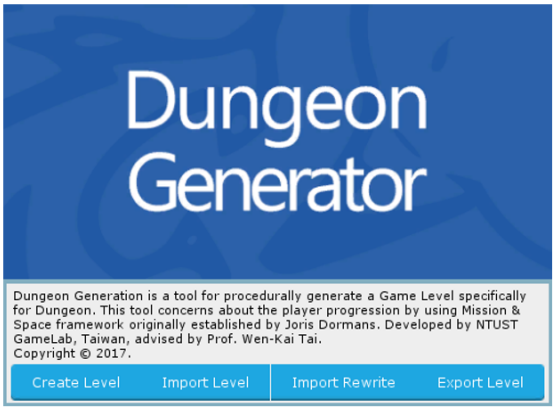

圖 3.2: Dungeon Generator 工具

3.1.1 建立任務符號表

在本小節，會說明在 Dungeon Generator 工具中如何操作符號表，進行新增、修改與刪除等動作；以及解析 **The binding of Issac** 遊戲關卡，萃取其遊玩特徵並作為任務語法的節點。

在任務語法中，任務節點可表示表示為一項事件、挑戰、動作或遊戲物件等。

萃取遊玩特徵

然而此步驟通常與下一步驟「建立任務規則」同時進行，.....。

任務符號表使用說明

content here.

圖 3.3: 利用 Dungeon Generator 建立符號表

3.1.2 建立任務規則

任務規則分為左側規則及右側規則，二者皆為圖形語法 (Graph Grammars) 所構成，左側規則為被取代方、右側規則為取代方，在任務圖中若有子圖符合任務規則的左側，將會依照流程進行替換改寫動作至規則右側。為了方便管理與分類不同性質的規則，在 Dungeon Generator 中定義一任務語法包含了多個任務規則群組，各自群組底下有複數個任務改寫規則。規則被賦予使用數量上的條件限制，分為使用上限次數與下限次數，上限次數表示完整的改寫系統運行過程，最多能夠套用該規則的次數上限，多半是為了因應部分遊玩特徵的獨特性或稀有性；反之，下限次數表示套用該規則的次數下限，此外能利用於改寫系統的疊代結果已趨於穩定，任務圖無剩餘任何非終端節點的情形下，作為持續進行改寫系統的強制約束條件。

線性任務規則

玩家在進行遊戲時，會有一條以上的主要任務流程、劇情安排，

遊戲設計師能夠依照需求定義出無窮的疊代規則，舉例來說，

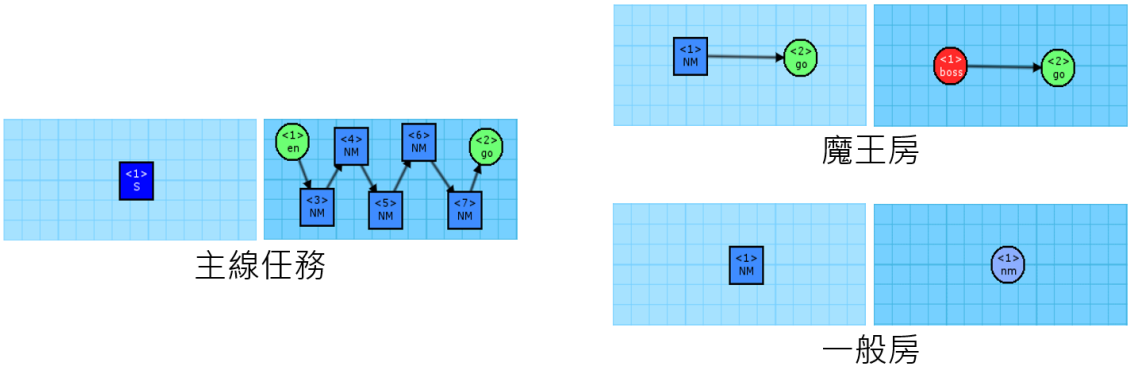

圖 3.4: 線性任務規則範例

非線性任務規則

為了要構築出

Image is under construction

圖 3.5: 非線性任務規則範例

排除非法規則

設計規則時，Dungeon Generator 將排除不合法的九項設計原則，非法的設計特徵會導致改寫系統出現錯誤，例如：改寫系統陷入無限循環或任務圖破碎化。在

表 3.1 所列舉之條件於某些情況中，可能彼此會同時符合多項非法條件，系統將依照判斷順序擇一輸出。

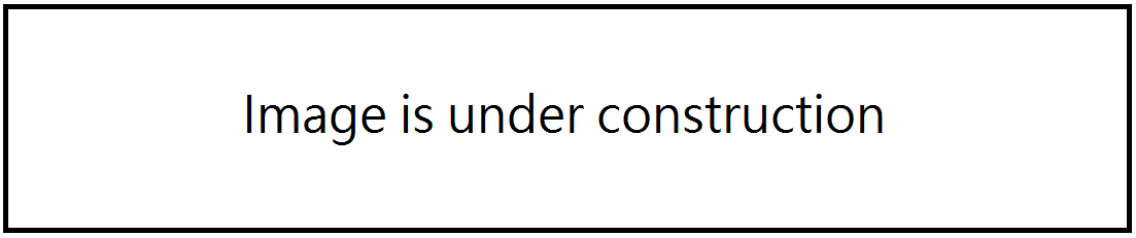

圖 3.6: 於 Dungeon Generator 工具中，若規則為非法狀態將不予生效

3.1.3 產生任務圖

根據 Joris Dormans [2] 提出的方法並修改至符合我方實驗需求，歸納出演算法 1。任務語法的改寫系統會以深度優先搜尋法 (Depth-first search) 遍歷整個任務圖，過程中遵循以下步驟順序。第一步驟，源節點會從任務規則集合中過濾出相符的規則，並從複數個規則中依照採輪盤法 (roulette wheel selection) 選擇出一項任務規則稱之為匹配規則 (matched rule)，並從源節點進行與匹配規則的改寫替換，符合的條件為定義遍歷中的源節點作為新圖，而新圖存在一個子圖與匹配規則的左側之圖形語法同構，同構的參考基準為節點的符號種類，在確認子圖同構的同時，有關連的節點會標記與匹配規則一致的索引值。第二步驟，將任務圖含有索引值的節點，其相連的連接線移除。第三步驟，任務圖含有有索引值的節點取代成匹配規則右側的等價節點。第四步驟，將匹配規則右側中沒有與左側等價的節點添加至任務圖中。第五步驟，依照匹配規則右側的連接線，以相同方式放入任務圖中。最後一步驟，將任務圖中的索引值移除。

Algorithm 1 RewriteSystem1 - 任務語法的改寫系統

```
if matchedRule is found from root then
    matchedRule = FindMatchs(root)
    移除與 matchedRule 相符子圖的連接線
    將相符子圖的節點，依照對應索引值替換成 matchedRule 的規則右側
    將 matchedRule 剩餘的節點填充至任務圖中
    依照 matchedRule 的連接情況，移轉到任務圖中
    移除節點的索引值相關資訊
end if
for all child = root->children do
    RewriteSystem1(child)
end for
return root
```

Algorithm 2 利用 VF Graph 進行子圖同構的搜尋

```
return 編輯中
```

表 3.1: 非法的任務規則定義

非法規則之標籤	標籤的狀況描述
LeftMoreThanRight	當左側的節點超過右側的節點數量時，進行改寫系統會使左側無法對應到右側的節點產生遺失的情形。若這些節點原先已有與其它非規則內節點連接，將會導致該連接資訊遺失，有機會造成任務圖破碎。
EmptyLeft	左側為空將無法進行子圖搜索，因此左側節點必須至少一個節點。
IsolatedNode	孤立的節點將會導致任務圖破碎。
IsolatedConnection	孤立的連接線無法正確表示其連接資訊，將無法正常進行改寫系統。
ExactlyDuplicated	若左右規則同構將導致改寫系統陷入無限循環。
MultipleRelations	兩兩節點間不可有超過一個的連接關係，不論是同向連接線或反向連接線皆會導致改寫系統無法正常運作。
CyclicLink	任務圖的定義中，任務應嚴格遵守任務間之順序性，若有循環結構將會使玩家迂迴停滯。
OrphanNode	若有圖形語法含有兩個以上的根結點，便無法正確定位出任務起點。
OverflowedAnyNode	當右側規則使用 Any 節點時，其對應到左側索引值的節點亦必須為 Any 節點。反之，左側規則使用 Any 節點將不在此限。

3.2 房型建構

Joris Dormans 於文獻中提到為二維空間的範例，我們的實驗環境以三維空間為主。在空間語法中將直接構築遊戲的基礎房型，但不設置怪物、寶箱或陷阱足以直接影響遊戲性的遊戲物件，如圖 3.7。此外，我們希望空間中的遊戲物件能夠有意義的自動化配置，即在設計空間語法的流程中，忽略絕大部分的遊戲物件配

置，直到 3.3 節提出之方法達成。

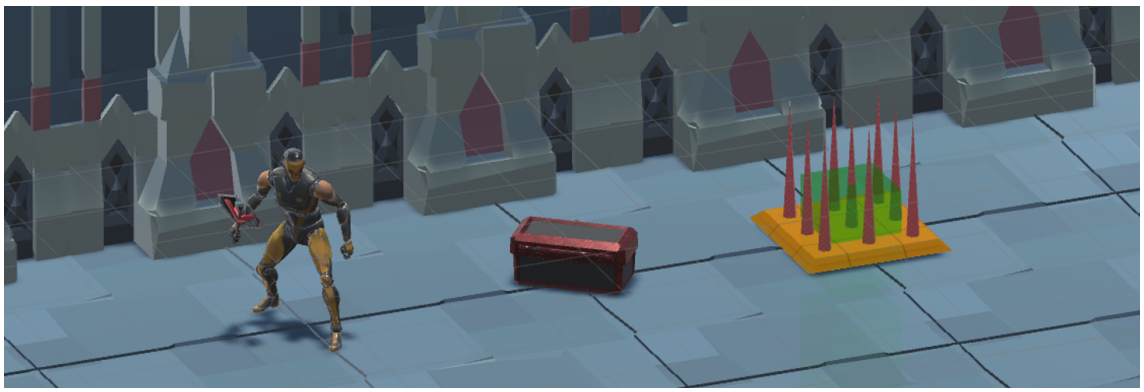

圖 3.7: 能夠直接影響遊戲性的遊戲物件，分別為敵方、寶箱與陷阱

我們對於空間語法做了修改以利實驗環境建置。圖 3.8 所示，在一個關卡 (level) 中包含數個房間容器 (volumes)，每一房間由不定數量的房間塊 (chucks) 組成，且房間塊固定以 9x9x9 個長方體體素 (voxels) 所構成，每一體素的大小為 3x2x3。為了要從已生成完畢的任務圖再衍生出分支的遊戲空間，在創建的房間容器會對應一任務語法之符號表當中一的終端符 (terminal symbols)，這樣的關係稱作為建造指示 (building instruction)；若在改寫規則運行中，且有多項規則同時符合替換的條件時，系統會基於它們的關聯權重 (relative weight) 隨機挑選一個規則。

Image is under construction

圖 3.8: 房型之結構圖

此外，每一個體素被分為九種方向，分別為正四方、斜四方與中心點。不同的裝飾物會依照其特性決定能夠放置的方位，如牆壁放置於正四方；牆壁柱放置於斜四方；地面或階梯則放置於中心點。在一個體素中，多個裝飾物是能夠同時並存的，例如在其放置地面、一道牆壁、兩道牆壁柱。在本次實驗環境中，我們將會使用圖 3.8 中，「地面、階梯、牆壁、牆壁柱、門」共五種裝飾物進行房型的建置

工作。

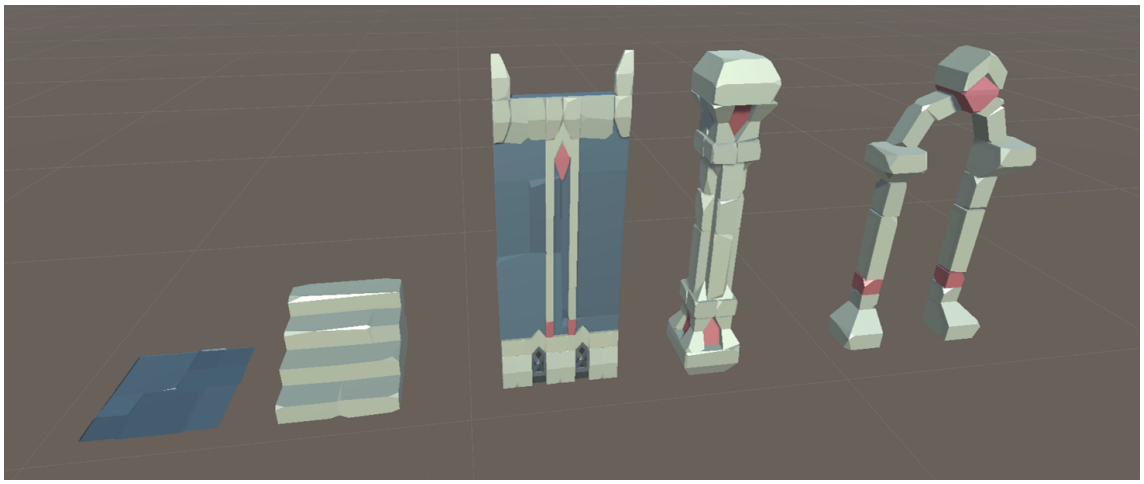

圖 3.9: 五種裝飾物放置於體素結構的情形示意

3.2.1 端點識別物

逐一設計出各式各樣的房型空間後，必須正確地將個空間相互連接。參考 Joris Dormans [2] 的空間語法，我們為體素型的房型添加端點識別物 (connections)，而端點可再細分為入口 (entrance) 與出口 (exit)，並且依照遊戲設計師需求，能夠自行擴充出口的類型，為複數個出口識別物 exit-name（exit-A、exit-B）等。在一房型空間中，最多僅能擁有一個入口識別物，而出口識別物的數量不在此限，且二者之數量總和必大於零。倘若空間當中沒有一個以上入口識別物，便會從現有的出口識別物中隨機挑選一與入口識別物相同功能執行之。

3.2.2 房型類型

針對實驗採用的遊戲主題，我們依照房間性質，區分出數項不同房型的空間規劃。

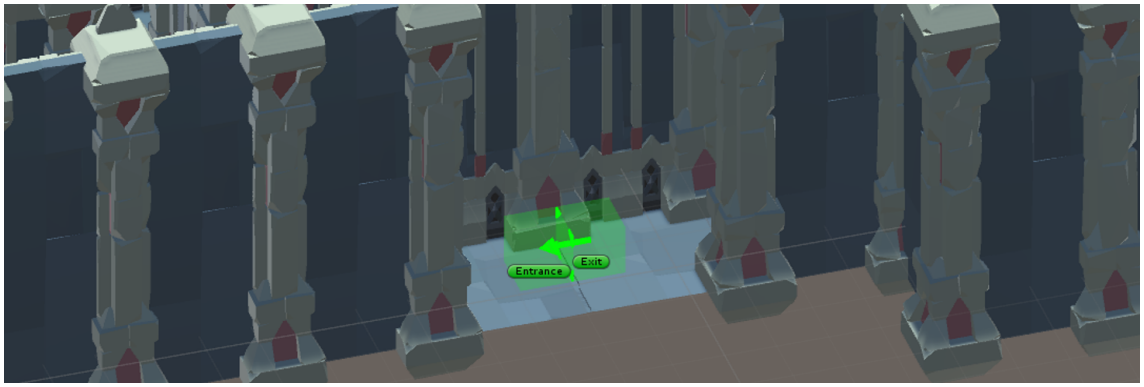

圖 3.10: 端點識別物的實際使用與串接情形

入口

入口為玩家的起始房間，若視空間為圖狀結構，。

Image is under construction

圖 3.11: 房間類型 - 入口

出口

content here.

Image is under construction

圖 3.12: 房間類型 - 出口

通道

content here.

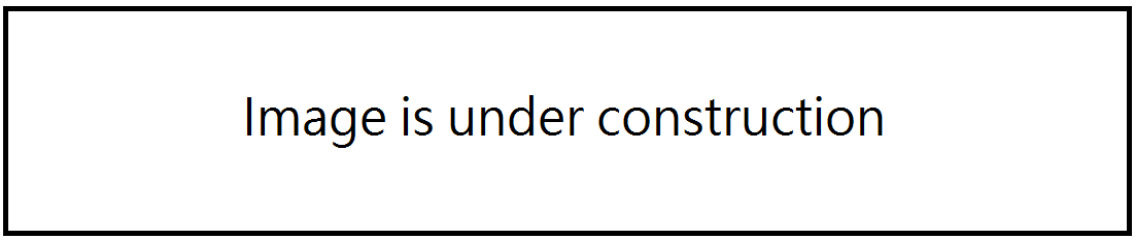

圖 3.13: 房間類型 - 通道

探索房

content here.

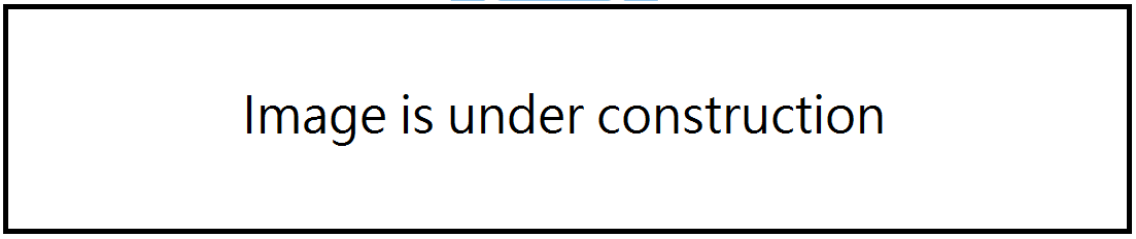

圖 3.14: 房間類型 - 探索房

戰鬥房

玩家經過戰鬥房時，必會與敵方發生戰鬥衝突，玩家必須設想戰術策略方可擊退敵方單位。

獎勵房

於獎勵房中，玩家能夠獲得價值不等的寶物道具，但寶箱多有敵方單位在周圍進行守衛。

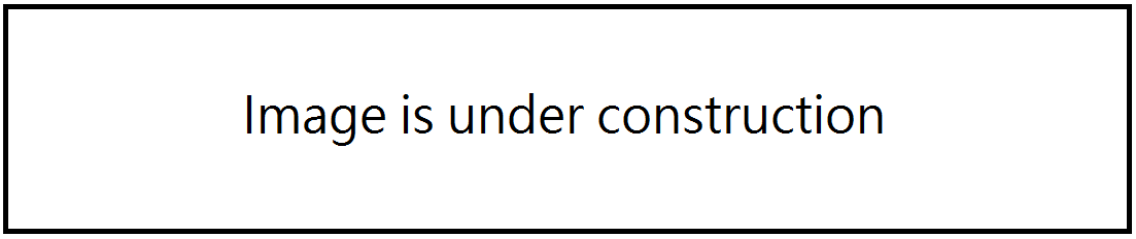

圖 3.15: 房間類型 - 戰鬥房

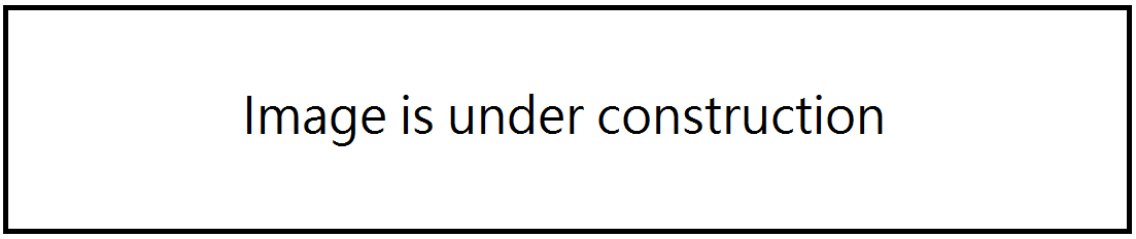

圖 3.16: 房間類型 - 獎勵房

3.2.3 從任務圖轉為遊戲空間

Algorithm 3 任務轉為空間的改寫系統

if some condition is true **then**
  do some processing
else if some other condition is true **then**
  do some different processing
else
  do the default actions
end if

3.3 地圖片段

延續上一小節的關卡結構，房間視為單一染色體 (chromosome) 之個體 (individual)；房間中能夠放置遊戲物件的各點座標視為基因 (genes)，而基因的類型有空磚、怪物磚、寶箱磚與陷阱磚等；同一個房間會擁有多種不同遊戲物件配置情況

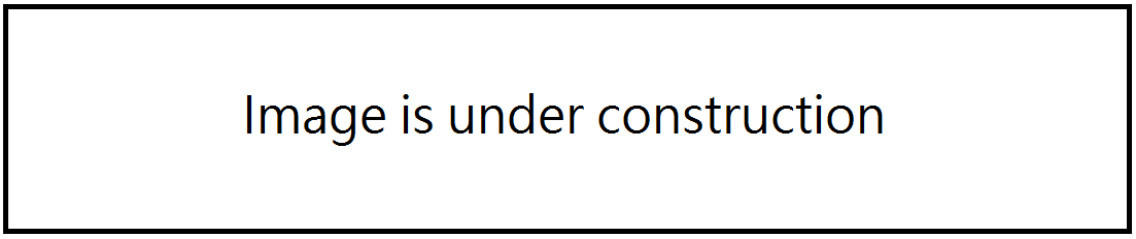

圖 3.17: 已疊代生成的任務圖，轉換為任務空間之結果

的染色體，這些染色體的集合便是族群 (population)。第一步驟，產生初始父母代族群時，讓全部的基因先預設為空磚，並使各染色體先行隨機突變；第二步驟，透過適應性函數計算各染色體的適應值 (fitnesses)，完整的適應性函數在 3.3.1 小節中說明；第三步驟將會從族群中挑選最優異的兩個父母染色體，高機率進行交配，若無進行交配將會將子代沿用父母代的基因，採用的交配方法在 **??** 小節中說明；第四步驟有低機率讓衍生的子代進行突變，不同的突變方式於 **??** 小節中說明；第五步驟以新的衍生子代取代舊有的父母代族群；第六步驟會檢查是否達到終止條件，若尚未滿足終止條件，便會回到第二步驟，直到輸出最佳解。

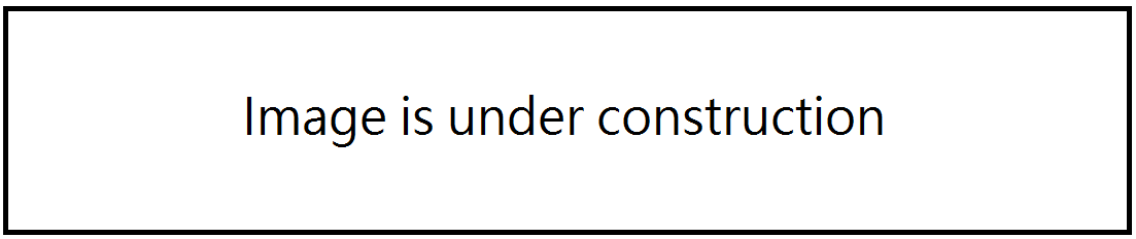

圖 3.18: 地圖片段採取基因演算法之流程

3.3.1 演化之適應性函數

動作冒險遊戲 (A-AVG)、動作角色扮演遊戲 (A-RPG) 等類型遊戲，多可見一些制式化遊戲物件的搭配組合，我們嘗試汲取出多項遊玩特徵並參數化公式，作為評估關卡品質的指標之一。在前處理時，我們使用 A-Star 演算法尋找入口至多個出口的最短路徑，凡經過的座標稱作為空間動線 (MP) 之一，並將其權重值 (mp) 增

加一，空間動線為多項指標關鍵性的參考依據。

守衛點 **(Guard)**

為體現出敵人會保衛寶箱 (T) 與出口 (E) 的現象，計算敵人 (E_i) 與關鍵性較高遊戲物件 (O_j) 之間的距離，倘若距離愈近則帶來的影響力愈大。

$$f_{grd} = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{1}{dist(E_i, O_j)}, O_j \in \{Treasure, Exit\} \tag{3.1}$$

阻攔點 **(Block)**

敵人會專注於阻擋玩家繼續前進，迫使玩家與其發生衝突。敵人會配置於動線上，mp_i 為空間動線權重；e_j 為該敵人於空間之動線權重，倘敵人並未落在動線上，則該項為 0。

$$f_{blk} = \log_{\sum_{i=1}^{N} mp_i} \sum_{j=1}^{M} e_j \tag{3.2}$$

攔截點 **(Intercept)**

與阻攔點近似，但敵人會被配置於動線附近非動線上，以快速追擊玩家為目的。各敵人 (E_i) 越接近空間動線各點 (MP_j) 時影響愈大，且動線權重 (mp_j) 亦會影響加權程度。

$$f_{itc} = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(\frac{1}{dist(E_i, MP_j)} \times mp_j \right), E_i \neq MP_j \tag{3.3}$$

巡邏點 **(Patrol)**

確保各敵人擁有足夠的空間能夠進行移動。將計算敵人 (E_i) 與指定半徑 (r) 內的座標數量 (P_j) 總值，當中並不包含不可通行的牆壁等類型。於本次實驗中，我們將採用 $R = 3$ 作為實驗範例，該數值可由遊戲設計師決定。

$$f_{ptl} = \sum_{i=1}^{N} \Big(d_i \times \sum_{j=1}^{M} count\big(E_i, P_j\big) \Big), d_i = \begin{cases} \frac{1}{2^i}, & \text{if } i \neq N \\ \frac{1}{2^{i-1}}, & \text{if } i = N \end{cases} \tag{3.4}$$

$$dist\big(E_i, P_j\big) \leq r, P_j \notin wall$$

至高點 **(Dominated)**

當玩家可能所在動線上之位置 (MP_j) 與敵人的位置 (E_i) 具有高低差時，敵人便適合採取遠程攻擊；為了提供玩家思考對付遠程敵人的緩衝時間，將敵人配置於動線末端附近是較好的選擇，j 隨著動線的順序演進，影響程度逐漸增幅。

$$f_{dom} = \sum_{i=1}^{N} \sum_{j=1}^{M} \Big(\frac{1}{dist\big(E_i, MP_j\big)} \times mp_j \times j \times high(E_i, MP_j) \Big) \tag{3.5}$$

支援點 **(Support)**

敵人 (E_i, E_j) 之間擁有一定程度的護援關係，當敵人彼此的距離愈低其影響程度越大，同時該敵人 (E_i) 必須遠離動線 (MP_k)。

$$f_{sup} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{1}{dist(E_i, E_j)} + \frac{1}{M} \sum_{k=1}^{M} \frac{1}{dist(E_i, MP_k)} \right) \tag{3.6}$$

死角點 **(Neglected)**

由於房與房之間的牆壁阻隔，使得敵人能夠埋伏於入口附近之死角處，出奇不意地對玩家展開攻擊。為了體現出這種現象，我們將敵人 (E_i) 與主要動線上各點 (MP_i)，兩端點連線之對角線所構成的立方體，立方體所涵蓋各座標點 (N_j) 至該對角線的距離為 d_k，隨著距離增加影響程度會衰減；vis_k 為該點的可視情形，若有不可視的座標存在便會提高適應值。隨著動線的順序演進，影響程度逐漸衰減。

$$Underconstruction \tag{3.7}$$

3.3.2 負數權重的適應性函數

在 3.3.1 守衛點指標的設計中，當權重為正數時能夠體現出「需要被守護的物件周圍出現敵方單位進行守衛」的遊玩特徵，且在多個物件的情形下儘可能平均分配敵方單位；反之，負數權重便會體現出「愈不平均數量的守衛情形」。緣故為設計守衛點指標時，並無針對寶箱的數量進行控管而導致。

3.3.3 多項適應性函數合併

在設計適應性函數初期，為求多個適應性函數相互牽制，進而求得限制條件的最佳解。而初期函數會力求場上所有的敵人必須盡可能符合各項指標，舉例來說，在某一房型的設定中，我們將守衛點、阻攔點的權重調整至 1，其餘指標將不採計得分即不調整權重，倘若場面上存在著二個敵人 A 與敵人 B，他們會同時被設置在動線上且一定距離內會有作為守護對象的寶箱物件。但對於遊戲設計師在某些情形下，敵人 A 與敵人 B 僅需要分別符合守衛點、阻攔點指標，才是理想中的遊玩特徵。因此，我們定義首次出現相關遊玩特徵的得分將會最高，並依照數量逐漸下降成長幅度且逼近於 1，呈現近似於底數大於 1 的對數函數圖形。藉著取

代原先的線性成長關係以外，便可以達到數值的正規化，強制將值域上限規範至 1 以求最後適應性函數加權總分時，不偏袒任何適應性函數以達到公平基準。

然而在對數函數中，當真數為 1（符合指標的敵人數量為 1）時，所得之值為 0 並不符合我們的訴求。

3.3.4 套用物件演化機制的房型空間

綜合上述小節，不同房型類型會有對應的戰略配置，因此依照房型類型設置不同的適應性函數權重

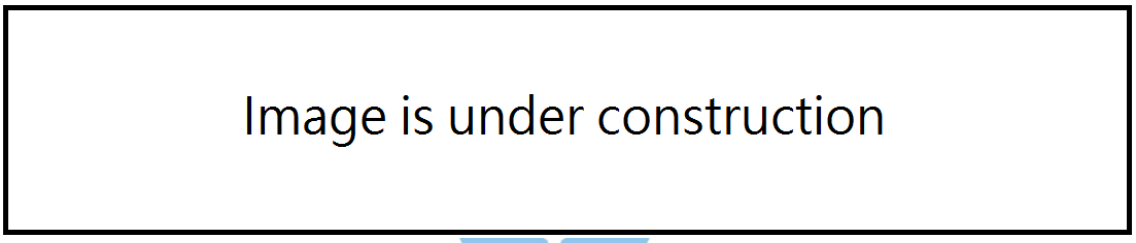

圖 3.19: 將房型空間作為胚胎，進行空間內遊戲物件的演化生成

第 4 章　實驗結果與分析

為了探討前述方法是否符合需求目標，將進行以下實驗： 4.4 小節中，針對所挑選的房間採用不同權重的適應性函數，會如何影響房間內的遊戲物件之配置結果。 4.5 小節中，觀察染色體中的基因數量對於演化過程之影響。 4.6 小節，進行評估不同的交配策略下，何者優劣優勢之情形。

4.1　實驗定義

在任務語法階段時，我們將非合法... (編輯中)

地圖片段演化階段時，每一次世代的演化過程，有 80% 機率父母代間會進行兩點交配 (two-point crossover)；10% 機率衍生子代會進行突變，染色體個體中有 5% 至 20% 的基因數量會轉換成其它的物件種類。

4.2　資料收集

在資料收集階段中，將收集 3.3 基因演算法於各實驗、世代與其個體（單一染色體）中基因類型的得分狀況。

4.3　各世代演化狀態

由於各點基因，其中的空格、敵人兩種類型，於每一遊玩特徵的指標都具有高度意義與相關性。

表 4-1: 原始資料之欄位

實驗編號	世代編號	染色體編號	指標	得分	座標	類型	房間
1	1	1	Block	0	(12.0, 1.0, 0.0)	Empty	Room A
1	1	1	Intercept	0	(12.0, 1.0, 0.0)	Empty	Room A
1	1	1	Patrol	0	(12.0, 1.0, 0.0)	Empty	Room A
1	1	1	Guard	0	(12.0, 1.0, 0.0)	Empty	Room A
1	1	1	Support	0	(12.0, 1.0, 0.0)	Empty	Room A
1	1	1	Block	0	(24.0, 1.0, 12.0)	Empty	Room A
1	1	1	Intercept	0	(24.0, 1.0, 12.0)	Empty	Room A

4.4 演化結果與其品質

content here.

4.5 房型規模之比較

房型的大小較有可能直接影響和可行走瓦磚之數量。本階段的實驗中，我們提取 4.2 節的資料，將空白、敵人兩種類型的數量關係繪製成熱圖進行觀察。這是因為各項適應性函數在設計時，多以「敵人」與其餘敵人、其它遊戲物件或玩家動線為考量參考，因而推估二者間勢必存在者某些關係。

4.6 評估交配策略優勢

content here.

第 5 章　結論與後續工作

5.1　貢獻與結論

我們提出將抽象化的動線作為評估的指標，驗證即使精確度降低的情形下，仍確保結果具有一定品質。(編輯中)。

5.2　限制與後續工作

編輯中。

參 考 文 獻

[1] J. Chen, “Flow in games (and everything else),” *Communications of the ACM*, vol. 50, no. 4, pp. 31–34, 2007.

[2] J. Dormans, “Adventures in level design: generating missions and spaces for action adventure games,” in *Proceedings of the 2010 workshop on procedural content generation in games*, p. 1, ACM, 2010.

[3] J. Dormans, “Level design as model transformation: a strategy for automated content generation,” in *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, p. 2, ACM, 2011.

[4] J. Dormans *et al.*, *Engineering emergence: applied theory for game design*. Creative Commons, 2012.

[5] A. Liapis, G. N. Yannakakis, and J. Togelius, “Generating map sketches for strategy games,” in *European Conference on the Applications of Evolutionary Computation*, pp. 264–273, Springer, 2013.

[6] A. Liapis, “Multi-segment evolution of dungeon game levels,” 2017.