



國立台灣科技大學  
資訊工程系

---

## 碩士學位論文

我的碩士論文題目

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

論文摘要	I
Abstract	II
誌謝	III
目錄	IV
圖目錄	VI
表目錄	VII
1 緒論	1
1.1 迷宮探索遊戲 (dungeon crawl) 類型介紹	1
1.1.1 迷宮探索遊戲的歷史	1
1.2 研究動機	1
1.3 研究目的	1
1.4 預計研究貢獻	1
1.5 本論文之章節結構	1
2 相關研究	2
2.1 含有敘事脈絡的關卡	2
2.2 關卡生成 (Level Generation)	2
2.2.1 程序化生成任務內容	2
2.2.2 程序化遊戲物件擺放	4
2.2.3 其它程序化生成技術	5
3 研究方法	6
3.1 任務語法	6

3.1.1	建立任務字符表 . . . . .	7
3.1.2	建立任務規則 . . . . .	7
3.1.3	產生任務圖 . . . . .	7
3.2	房型建構 . . . . .	7
3.3	地圖片段 . . . . .	8
3.3.1	各項適應性函數 . . . . .	8
3.3.2	不同交配方式 . . . . .	10
3.3.3	不同的突變方式 . . . . .	10
3.3.4	終止條件的設立 . . . . .	10
4	實驗結果與分析 . . . . .	11
4.1	實驗定義 . . . . .	11
4.2	資料收集 . . . . .	11
4.3	各世代演化狀態 . . . . .	12
4.4	Section C . . . . .	12
4.5	演化結果與其品質 . . . . .	12
4.6	房型規模之比較 . . . . .	12
4.7	評估交配策略優勢 . . . . .	12
5	結論與後續工作 . . . . .	13
5.1	貢獻與結論 . . . . .	13
5.2	限制與後續工作 . . . . .	13
	參考文獻 . . . . .	14
	授權書 . . . . .	15

# 圖 目 錄

圖 2.1	Antonios Liaps 提出的兩階段式關卡演化。 . . . . .	5
圖 3.1	本論文提出系統之流程圖。 . . . . .	6



# 表 目 錄

表 4-1 原始資料之欄位 .....	11
---------------------	----





# 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 迷宮探索遊戲 (dungeon crawl) 類型介紹

### 1.1.1 迷宮探索遊戲的歷史

## 1.2 研究動機

迷宮探索

## 1.3 研究目的

## 1.4 預計研究貢獻

## 1.5 本論文之章節結構

## 第 2 章 相關研究

本章劃分成四小節，第一節

### 2.1 含有敘事脈絡的關卡

content here.

content here.

content here.

### 2.2 關卡生成 (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.

content here.

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 [1] [2] [3]；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 [1] [3]，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。

## Dungeon Crawl

content here.

content here.

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...

### 2.2.2 程序化遊戲物件擺放

content here.

content here.

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 [4]。在 Sentient Sketchbook 中，遊戲關卡設計師能夠以低分辨率、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稿演化，第二階段為地圖片段 (map segments) 演化 [5]。地圖片段的結構類似於地圖草稿，由  $N \times N$  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。

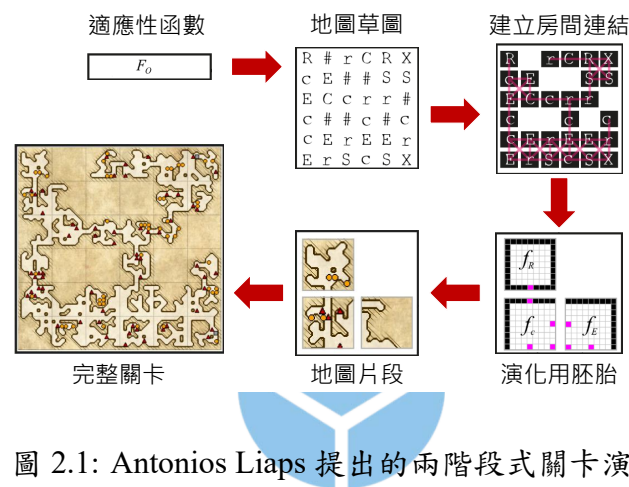


圖 2.1: Antonios Liapis 提出的兩階段式關卡演化。

### 2.2.3 其它程序化生成技術

content here.

content here.

content here.

#### 遊戲 - Spelunky

content here.

content here.

content here.

## 第 3 章 研究方法

本論文中，我們仍保持 Joris Dormans [1] 與 Antonios Liapis [4] 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，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巨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。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。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，採用基因演算法的過程，令各房間遵守適應性函數的限制得到符合訴求的最佳解。

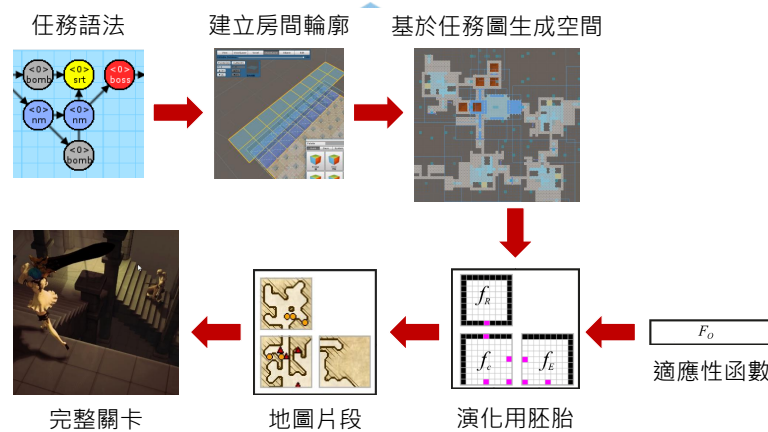


圖 3.1: 本論文提出系統之流程圖。

### 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。

在任務語法的設計階段，我們參考遊戲 Spiral Knight 的關卡地圖，分析其遊戲進程結構，繪製期望的任務圖並將觀察到的遊玩特徵，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。

### 3.1.1 建立任務字符表

### 3.1.2 建立任務規則

### 3.1.3 產生任務圖

---

**Algorithm 1** 任務語法的改寫系統

---

```
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
```

---



## 3.2 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。在空間語法中我們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 OOOOO。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。

我們對於空間語法做了修改以利實驗環境建置，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 (chunks) 組成，且房間塊固定以 9x9x9 個正立方體體素 (voxels) 所構成。為了要從已生成完畢的任務圖再衍生出分支的遊戲空間，在我們創建的房間容器會對應一任務語法之字符表當中一的終端符 (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。



### 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 3.3.2 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 3.3.3 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。



#### 3.3.1 各項適應性函數

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。

#### 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。

$$Underconstruction \quad (3.1)$$



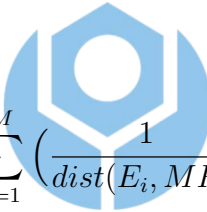
### 阻攔點 (Block)

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。

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

### 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。


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

### 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。

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

$$\text{dist}(E_i, P_j) \leq r, P_j \notin \text{wall}$$

### 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。

$$f_{grd} = \frac{1}{N \times M} \sum_{i=1}^N \sum_{j=1}^M \frac{1}{dist(E_i, O_j)}, O_j \in \{Treasure, Exit\} \quad (3.5)$$

### 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。

$$f_{dom} = \sum_{i=1}^N \sum_{j=1}^M \left( \frac{1}{dist(E_i, MP_j)} \times mp_j \times j \times high(E_i, MP_j) \right) \quad (3.6)$$

### 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 ( $MP_k$ )。

$$f_{sup} = \frac{1}{N} \sum_{i=1}^N \left( \frac{1}{N} \sum_{\substack{j=1 \\ j \neq i}}^N \frac{1}{dist(E_i, E_j)} + \frac{1}{M} \sum_{k=1}^M \frac{1}{dist(E_i, MP_k)} \right) \quad (3.7)$$

### 3.3.2 不同交配方式

### 3.3.3 不同的突變方式

### 3.3.4 終止條件的設立

## 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：4.5 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。4.6 小節中，觀察染色體中的基因數量對於演化過程之影響。4.7 小節，進行評估不同的交配策略下，何者優劣優勢之情形。

進行相關實驗時，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。

### 4.1 實驗定義

### 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。

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

### 4.3 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。

### 4.4 Section C

### 4.5 演化結果與其品質

### 4.6 房型規模之比較

### 4.7 評估交配策略優勢



## 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。

### 5.2 限制與後續工作

編輯中。



## 參 考 文 獻

- [1] 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.
- [2] 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.
- [3] J. Dormans *et al.*, *Engineering emergence: applied theory for game design*. Creative Commons, 2012.
- [4] 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.
- [5] A. Liapis, “Multi-segment evolution of dungeon game levels,” 2017.

