Adventures in Level Design: Generating Missions and Spaces for Action Adventure Games

摘要

本文研究動作冒險遊戲的關卡之生成策略。這種類型非常依賴良好的關卡設計，而非**規則主導 (rule-driven)** 類，像是策略、角色扮演這類在過去已經能成功生成關卡的遊戲。本文所提出的方法將**任務 (Missions)**與**空間 (Spaces)** 分為分開的結構，並各需要一個獨立的步驟生成。此討論了在程序中各個獨立的步驟下，各種不同**生成語法 (Generative Grammars)** 的優點。

關鍵字：程序化生成；關卡設計；動作冒險遊戲

目錄

其它

原文作者：

- Joris Dormans (j.dormans@hva.nl)

譯者：

- Hao-Chen Wang (bachelor.whc@gmail.com)
- Ze-Hao Wang (salmon.zh.tw@gmail.com)

審稿者：

- Kevin Lai (s900127127@gmail.com)
- Li-An Lin (jenny84311@gmail.com)

介紹

具有程序化生成內容的遊戲已經出現一段時間了。此類遊戲經典的例子是*Rogue*，一個老式龍與地下城風格的*ACSII*地城冒險遊戲，每當玩家展開新遊戲時關卡就會重新生成。近年程序化產生內容的遊戲原始碼往往是無法取得的，因此難以判別其使用於自動化生成的演算法為何。而老遊戲的原始碼與它們的關卡生成策略在網路上被良好地記錄著。這些遊戲典型的關卡生成方法可以被分為暴力亂數演算法，是為了此類遊戲結構而量身打造的方法。一個生成策略是生成格狀的地圖並代表其充滿岩石，並從地圖起點開始”鑽"出隧道和房間。複數的路徑可以由之前打造的房間分別朝新方向鑽出。接著便在該地城內放置生物、陷阱與寶藏。另一個策略採用將地城劃分為大區塊，接著在部分區塊中生成房間、最後將這些房間以迴廊連結起來[2]。如要生產代表荒野區域的遊戲空間，則使用**細胞自動機(cellular automata)**創造富有生機的結構。

雖然這些演算法的成效皆已在*roguelike*遊戲的關卡中被證實，但這類演算法生成出來的遊戲類型是非常少的。在*roguelike*遊戲玩法中，角色建設是一種典型的元素。此類型的遊戲玩法機械式地闡述紙筆桌上型角色扮演遊戲，解決大部分用以提升角色所需的收集的經驗點數、魔法裝備等繁複過程。就如遊戲設計師*Ernest Adams*在「來自地城的一封信」所諷刺地，此處在遊戲機制背後似乎缺乏了動機，導致用來描述角色成長的手法就如同*Mythical Quest*那微弱的回饋顯得不足[4]。此類型的遊戲玩法，雖然形成自己的利基，但非常適合隨機生成地城的佈局，並且不需要與關卡設計相同的質量，例如，動作冒險遊戲中薩爾達系列主設計師宮本茂在採訪中提到，角色開發僅作為遊戲的一小部分[5]。就如同隨機遇敵表是一個評價良好的工具並可以產生特定風格，但並非所有龍與地下城的角色扮演都可行[6]。

這有點類似*Kate Compton*與*Michael Mateas*所點出的，平台動作遊戲的關卡生成要難得多，此類遊戲的關卡設計是最關鍵的一部份[7]。動作冒險同樣也依賴能讓玩家在探索、心流體驗與故事性中感到愉快的關卡設計。這些原則便使得我們難以用*roguelike*遊戲中常見的演算法來解決關卡生成問題。這些演算法不能表達這些原則，因為比起地城個別房間、通道的觀點，通常更注重更大規模的操作。為了產生符合上述原則的關卡，我們必須改變我們的做法，基於這些原則的層次進行操作。而這方法，正是使用**生成語法(generative grammars)**。

然而，即便使用了生成語法，生成一個好的關卡仍非常困難。關卡往往讓我們有隨機感，使得整體欠缺一個結構。尋找單一生成語法用以擊敗所有難題是不可行的。良好的關卡設計，必非單一、而是有兩種結構；一個關卡總是存在著任務與空間。本篇論文建議，最好分開來使用生成語法生成適合各個結構的任務和空間。如此論文最終節所述，文中描述的方法將會先產生任務後再生成空間，並使任務分配至各空間中。

2. 任務與空間 (Mission and Spaces)

在由筆者所著作研究與描述*Zelda: The Twilight Princess* 遊戲的關卡 *Forest Temple* 中[8]，可見以兩種不同的結構都來描述關卡本身。首先，關卡中有幾何結構：空間。關卡可被歸納於由節點與連結組成的網路，來代表房間與其連結。再來，關卡中有一連串工作，必須由玩家完成方可抵達終點：任務。任務可以藉由**有向圖(Directed graph)**來表示前面有哪些工作需要完成。任務規定了工作完成的順序，並獨立自幾何結構。可在圖1看見任務能夠被配置到遊戲空間。在此例中，空間與任務的某些部分是**同構(isomorphic)**的。尤其是第一區的關卡任務和空間對應非常接近。任務與空間的**同構**在許多遊戲中常見，但任務與空間的結構差異通常卻相當重要

關卡空間配置了任務，而任務對應至特定空間，但他們卻是彼此獨立的。同樣一個任務可以對應多個不同的空間，而一個空間可以執行複數個不同的任務。任務與空間的設計原則又有所差別。一個線性的任務，所有的工作只可以單一、固定的順序完成，卻可以安排在非線性的空間配置中。同理，一個非線性的任務有許多平行的挑戰與選項，這些選項與挑戰能被安排在嚴格線性的空間中，讓玩家必須得來回奔波。

一些關卡的品質，可因任務和空間的功能而有所變化。例如，*Zelda* 的關卡與許多任天堂的遊戲常利用跟武術訓練相同的關卡結構，來使玩家熟悉特定的動作與技能[9]。以下所述的是一種結構，玩家在基本階段第一次學會了簡單的技巧，然後她在基本對練重複地使用練習這個技巧以達完美。在練習武術時，重複的動作可能是冗長又單調的；有個再好不過的例子，在電影《小子難纏》中，英雄一而再再而三重複執行同樣的動作直至完美。於是，在遊戲的對練階段玩家將學會不同的技術如何組合（對練階段），好讓她能在對抗頭目前熟練實技，這即是組手階段。這結構同樣適用於解析森之神殿的關卡。在這個關卡中，林克第一次學會了如何使用*bomblings* 來攻擊生物並解開通道（基本），他必須重複地使運用這個技術數次以推進關卡（基本對練）。他也會得到特殊道具迴旋鏢，並且同樣的，透過達成一連串簡單的目標（基本、基本對練）抵達終點。為了前往終點，林克必須要組合*bomblings* 與*boomerang* 來抓到最後一頭猴子，並必須到達神殿的最後一個房間，在那他也將使用相同的技巧解決關卡的最終頭目（組手）。

同時，森之神殿的任務也按照一些常常在好萊塢電影看到的劇情結構，如*Joseph Campbel*的著作*monomyth*一樣。（參見圖1. 在薩爾達傳說：曙光公主中，森之神殿的空間與任務[10] & [11]）依此結構，玩家必須對抗守門人才可跨越冒險境地（地城）。半路或二分之三的地方，玩家得擊敗中BOSS取得boomerang，這是第三幕開端的訊號；最終幕的結尾將是玩家擊倒關卡頭目。森之神殿的空間要素則有所不同，他的基本模式是軸心輻射式，這種結構讓玩家可以輕易地來回於神殿各個區域。*boomerang* 則作為一種鑰匙，在初期就能遇到並能打開許多機關。一旦在神殿內取得了*boomerang*，那麼神殿內額外的房間會解鎖並讓玩家可進行探索，此結構常見於冒險遊戲[12]。

3. 生成語法 (Generative Grammars)

生成語法源自於語言學，並作為於描述片語集合的模型。理論上，建立生成語法便可以從中產生所有正確的語言片語。一個生成語法通常由**字母表 (Alphabet)** 與**規則 (set of rules)** 所組成[13]。字母表是語法中所使用到的符號之集合；規則則採用改寫的方式進行，一項規則能夠指定什麼樣的符號（組）會被其它的符號（組）給替代，而形成其它的新字串。舉例來說，語法當中的規則可以指定符號裡的一段字串，將符號「S」替換成符號組「ab」，而這樣的規則下一般可以用「S→ab」來表示。

生成語法通常使用箭頭右側的符號（組）用以取代箭頭左側的符號（組）。因此，通常待替換的符號稱為**規則左側 (Left-hand side of the rule)**；並將新符號稱為**規則右側 (Right-hand side of the rule)**。字母表中的一些符號因為沒有指定替換的規則而從不被替換，這些符號稱為**終端 (terminal)** 並依照慣例以小寫字元表示。在前述範例的「a、b」即為終端符號，而**非終端符號 (Non-terminals)**指定他們依照規則進行替換並且通常用大寫字元表示等，先前範例中的符號「S」便是之一。對於描述自然語言句子的語法來說，終端符號可以是文字，而非終端符號表示功能性的文字組合，例如名詞片語和動詞片語。「S」時常被作為語法的起始符號一生成語法至少需要一個符號被置換，而不能從無開始。因此，完整的生成語法亦制定了起始符號。

像這樣的語法用於計算機科學以**創造語言**與代碼解析器，它們被設計為理解和識別語言。然而，語法也適合於**生成語言**。很容易見到看到簡單的規則能夠產生出相當有趣的結果，特別是當這些規則允許遞迴時：當規則產生的非終端符號，直接或是間接地導致相同規則產生。「S→abS」是遞迴規則中的範例，將會產生出無止盡的「ab」字串。「S→aSb」是另一個範例，並生成字串的「a」會隨著後方相等數量的「b」。為了讓自然語言所開發的生成語法，能夠領悟超越單一單字的水平，例如參數構造與修辭。這意味著利用生成語法來開發遊戲能夠找到更高級別的設計方法，進而在微觀與宏觀中引出有趣的關卡。

當以下的情形符合時，生成語法可被用來描述遊戲。首先，語法的字母表由一系列的符號組成，用以表示遊戲當中的特定概念。以及，規則定義了合理的方法，在前述概念能夠被組合後創建出良好的關卡。舉例來說，一語法描述冒險遊戲其可能的關卡，可能包括了終端符號「key」、「lock」、「room」、「monster」、「treasure」。而該語法的規則可能包含下列所述：

1. Dungeon → Obstacle + treasure
2. Obstacle → key + Obstacle + lock + Obstacle
3. Obstacle → monster + Obstacle
4. Obstacle → room

在上述的列舉中，當多項規則指定以*相同的非終端符號*進行可能性的替換，僅只會有一項規則會被套用。這可以用亂數隨機地進行，並能夠產生各式各樣的字串，結果可能包括了:

1. key + monster + room + lock + monster + room + treasure
2. key + monster + key + room + lock + monster + room + lock + room + treasure
3. room + treasure
4. monster + monster + monster + monster + room + treasure

上述討論的語法所產生的字串並非都適合遊戲關卡，特別是上述第三個字串過短。問題不是採用了生成語法，而是在舉例中使用規則的品質、質量。事實上，生成語法可以透過建立更好地抓住關卡設計原理的規則，來輕鬆地解決這些問題，像是:

1. Dungeon → Obstacle + Obstacle + Obstacle + Obstacle + treasure
2. Dungeon → Threshold Guardian + Obstacle + Mini-Boss + reward + Obstacle + Level-Boss + treasure

如果規則一包含了一個想法，一個地下城需要有一個最小長度的語法會是有趣的。規則二直接結合 **three act story structure** 來描述了 *Zelda: The Twilight Princess* 遊戲中的關卡 *Forest Temple*。

生成語法可以透過不同的方式來生成遊戲內容，遊戲專家與設計師可以產生語法來為特定的遊戲生成內容。製作這樣的語法不是件容易的事情，但是通過它來生成或調整新的內容所需的努力遠遠不及一開始。此外，語法和程序化內容能夠自動化設計一部分任務來輔佐設計師，而這方法被遊戲公司 Epic Games 用於生成建築物與大型城市景觀。它被證明是非常好用的方法，因為它允許設計師在相同的限制下，使用新的規則而不必重作整個部分，快速地重新生成先前幾個部分[14]。最後，可以從測試環境中選擇內容成功的**演化式演算法 (Evolutionary Algorithms)** 來生成語法。本文所提出的語法皆使用第一種方法來擬稿，演化式的語法雖然聽起來是個相當誘人的概念，但它超出了這裡所提及的知識範疇。

生成語法的相關應用也可以在 **Lindenmayer Systems (L-Systems)** 中發現。Lindenmayer 是一位生物學家，他利用了語法來描述植物的生長狀況，不過 L-Systems 已被應用在產生許多不同的空間輸出[15]。現今的 L-Systems 在遊戲中用於生成樹木和其它的自然結構。且 L-Systems 已經擴展到城市模型的程序化生成[16]。這項擴充用於創造循環的道路網路，其中原始的 L-Systems 僅生成樹狀結構。該擴充允許在先前所生成的街道附近再生成一街道並與之相交，因此創建出返回到之前產生的結構。

4. 以圖狀語法生成任務 (Graph Grammars to Generate Mission)

圖狀語法在 David Adams 的 2002 年學士論文— Automatic Generation of Dungeons for Computer Games 中討論關於關卡生成[17]。圖狀語法是生成語法的一種特殊形式，它不產生字串，而是由邊與點所組成的圖形。在圖形語法中，一個或多個節點以及相連邊可以由節點與邊所產生的新結構給替代[18]（請見圖二與圖三）。經過特定的規則所描述，將一組節點被選擇所替換之後，根據**規則左側**對所選擇的節點進行編號（圖三的第 2 步驟）。接下來，所有的被選擇的節點之間的邊將其移除（第 3 步驟）。有編號的節點接下來被取代成**規則右側**的等價節點（相同編號之節點）（第 4 步驟）。然後將**規則右側**中沒有與**規則左側**等價的節點添加至圖中（第 5 步驟）。最後，連接新節點的邊依**規則右側**放入圖中（第 6 步驟），以及移除編號（第 7 步驟）。請注意，圖狀語法可具有允許刪除現有節點的操作，但這些操作在本文中不被使用。

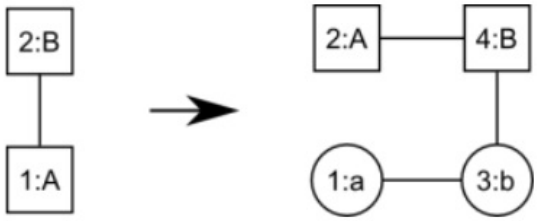

Figure 2. An example of a graph grammar rule

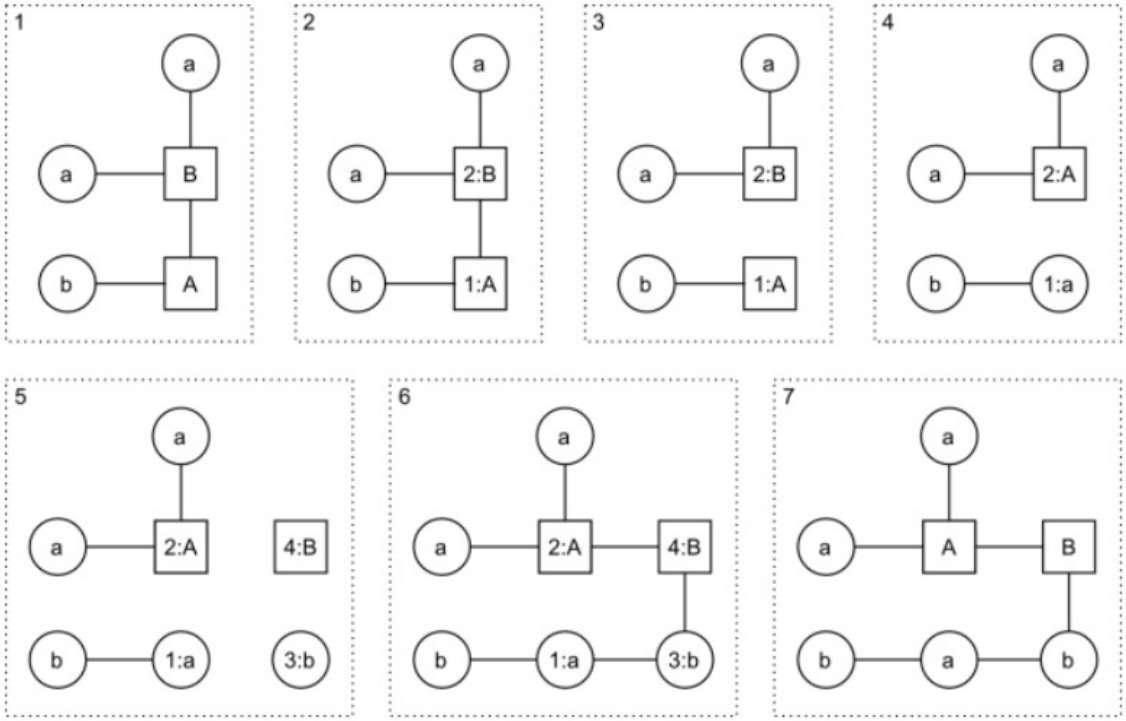

Figure 3. The replacement operations according the rules from figure 2.

圖狀語法非常適合生成任務，因為任務相當適合以非線性圖來表示。它會需要一個由不同的任務所組成的字母表，其中包括了挑戰與獎勵。圖四表明了一些規則，用於生成與 *Forest Temple* 任務類似的任務結構。圖五為生成語法的範例輸出。請注意，該語法包括了兩種類型的邊，一為單箭頭、二為雙箭頭，不同類型的邊是一項能夠在其它圖狀語法中找到的特徵。在這裡的情況，雙邊表示在從屬節點與其上級節點**緊密耦合 (tight coupling)**：這意味著從屬節點必須放置在生成空間中的上級節點之後，針對本文所描述的來具體實行。一般邊表示**鬆散耦合 (loose coupling)**，並代表從屬節點可以被放置在任何一處。

這對於生成空間的演算法來說非常重要（詳見下面第6章）

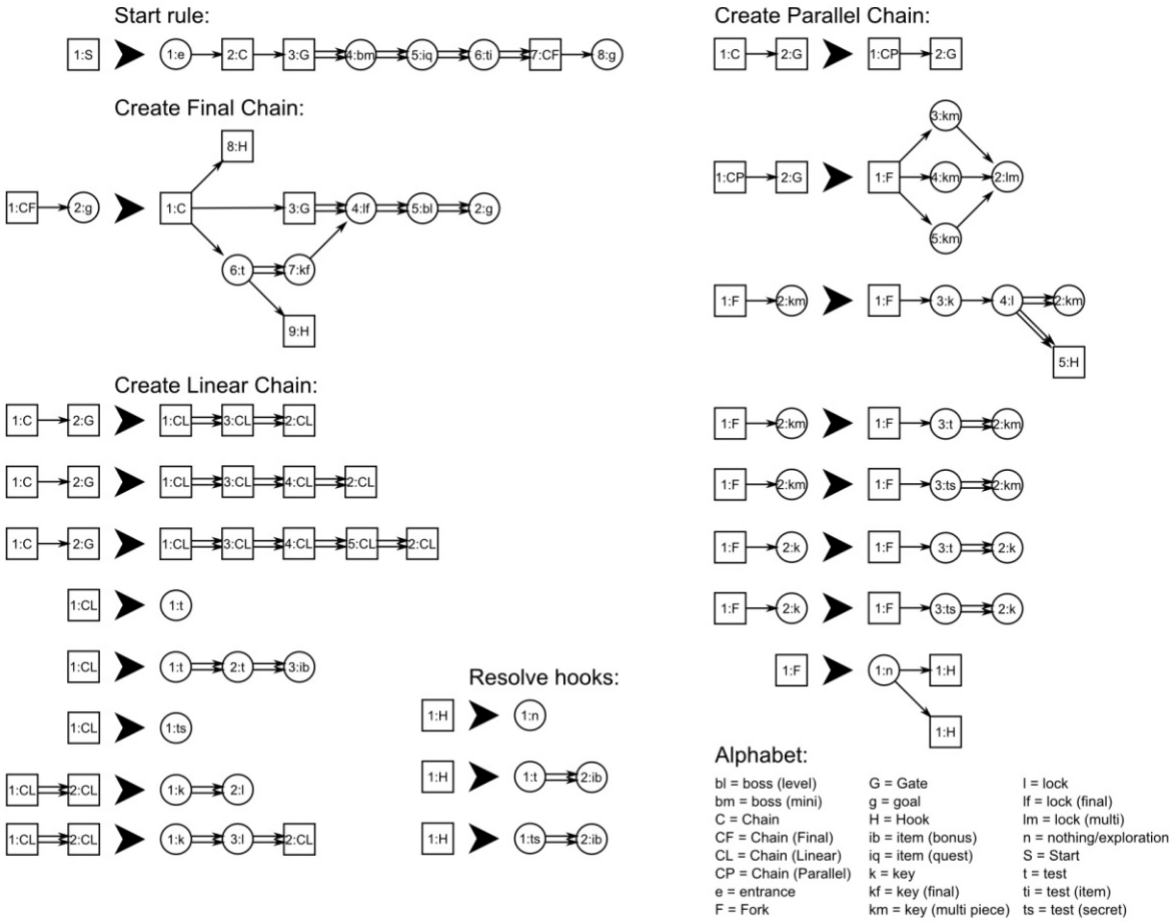

Figure 4. Rules to generate a mission

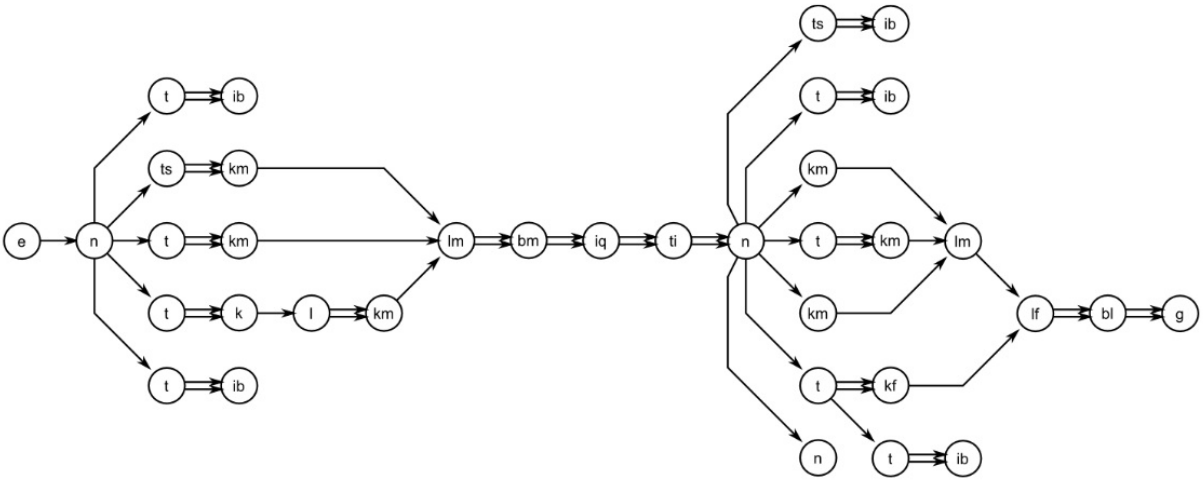

Figure 5. A generated mission (from the rules in figure 4)

5. 以形狀語法生成空間 (Shape Grammar to Generate Space)

形狀語法 (Shape Grammars) 對於生成空間是最有用的，自 1970 年初，形狀語法首次被 George Stiny 與 James Gips 所描述[19]。形狀語法類似於生成語法與圖狀語法，是根據重寫規則產生出的新形狀替代形狀。**特殊標記 (special markers)** 用於識別起始點或是協助定位（有時是縮放）新的形狀。

舉例來說，設想一形狀語法的字母表包含了三種符號：「a wall」、「open space」與「connection」（請見圖六-a）。在這一語法中，**僅有「connection」是非終端符號**，其帶有指向性三角形的正方形標記。在形狀語法**規則右側**標示的灰色標記，表示了原始形狀的位置與其指向為何。我們可以設計一些規則，讓 connection 會被取代成一小段的通道、T 字叉路或是替換成牆壁，進而有效地關閉 connection（請見圖六-b）。顯而易見的是，圖六-c 中所描述的結構便是上述規則可能的產出，只要起始符號也是 connection，並假設在每次的迭代時選擇隨機的 connection 來替代。

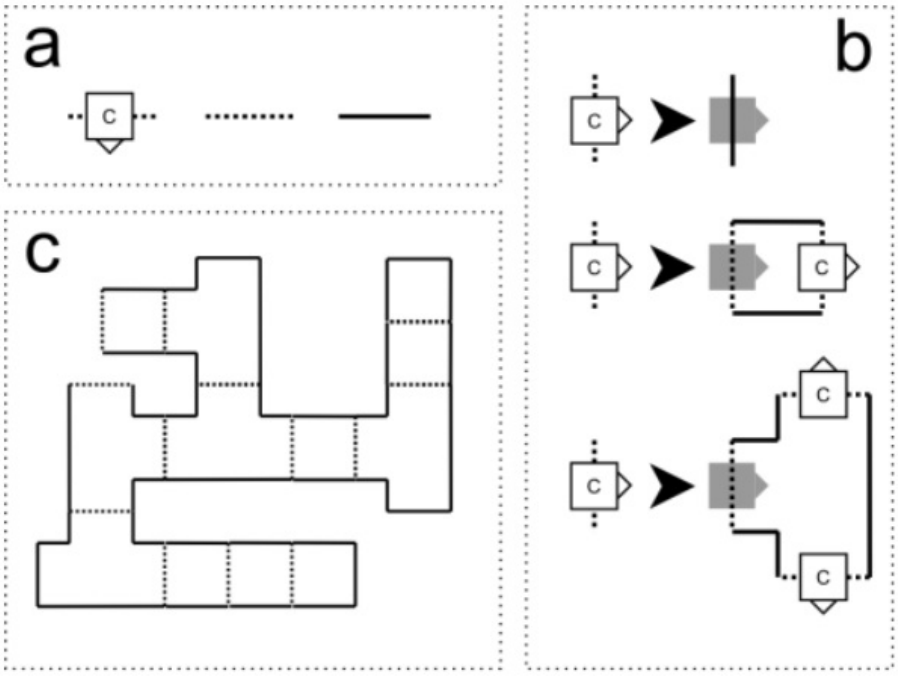

Figure 6. Shape grammar a) alphabet, b) rules and c) output

形狀語法像是任何的生成語法，包括可以遞迴。在結果的形狀中，遞迴引入了更多變化的好方法。舉例來說，在圖七中的規則是具遞迴性，以及這些規則產生出的形狀更具有自然（**碎形 fractal**）的感覺。在這種情形下，語法的實現應該允許**規則右側**可以被調整大小，來匹配正在成長形狀的大小。

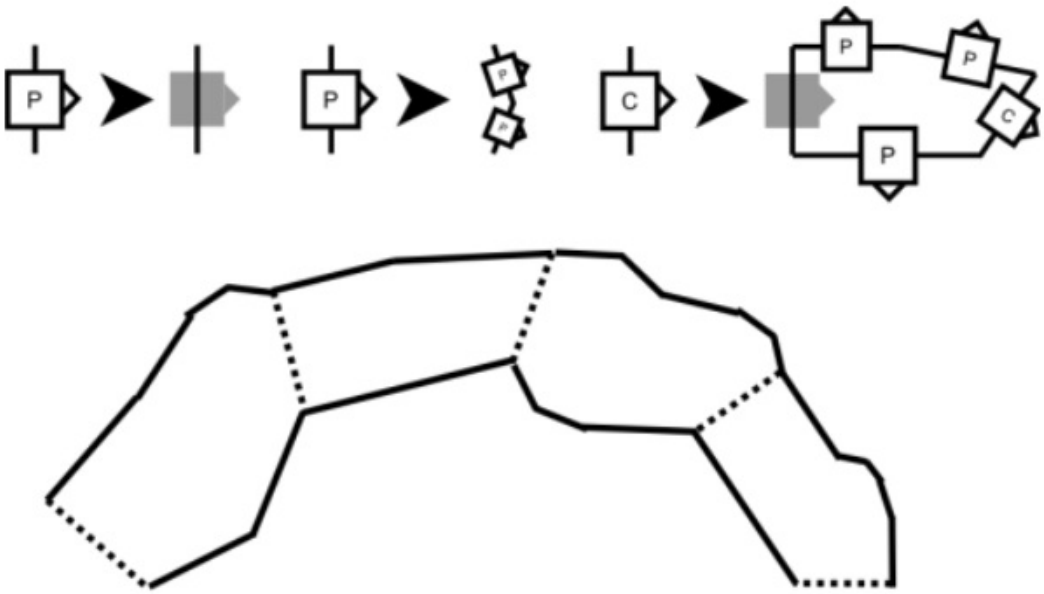

Figure 7. Recursive shape rules and output

6. 由任務產生空間 (Generating Space From Mission)

為了從已生成的任務中使用形狀語法(Shape grammar)產生出一個空間，我們對形狀語法(Shape grammar)做了些微的調整。任務的中止符(Terminal symbols)必須在形狀語法中作為建造指令(Building instruction)發揮作用。為了實現任務中止符在形狀語法中的機能，形狀語法的每個規則都得與任務語法的中止符有所關連。形狀語法首先會搜索任務中的下一個符號，接著尋找可實現此符號的規則並基於它們的關聯權重(Relative weight)隨機挑選一個規則；然後從可適用此規則的各種可能擺設位置中再根據關聯適用度(Relative fitness，該位置可能有其他更合適的選擇)隨機選擇其一。當每個任務符號的元素被生成後，這個演算法會儲存一個參照(Reference)至其之中，這讓演算法可實作出由任務控制的緊耦合(tight coupling)。此機制避免演算法把鑰匙、道具隨機擺放而非如任務規範地安排在測驗、門鎖其後。形狀語法更針對一些影響規則選擇的動態參數做了擴充。這些參數用於建立程序化難易度或是在不同註記(Register)之間的切換。舉例來說，這個語法在每次選擇規則時有更高的機率選擇更困難的阻礙，並將生成許多陷阱的註記替換為存在與多怪物的註記。

在應證此研究的測試程式裡，規則可以擁有與它們對應的指令。這些指令在規則被使用的前後皆可執行。它們促使動態的規則權重與程序化難易度等。

有關於形狀語法的額外補充是，此形狀語法靈感來自於可自動鋪設交叉口道路的L-System(請見Section 3)。為了保證空間成長時確實能與先前已生成的空間重新連結，因此演算法需要加入一個步驟。該步驟執行於一個已擺放於空間的規則之後，它會尋找兩個鄰近且能適當對齊的connection，並將兩個空間連結起來。為了避免任務出現短迴路，例如將最終房間與入口附近的房間意外地連結起來，所有在生成結構中開放的connection都必須在特定的規則執行前或後封閉。這些指令對應到以實作此類邏輯的規則。

一旦完整的任務都被考慮過以後，形狀語法便退回正常的實作，且會繼續迭代直到所有非中止符被一套可完善此空間的規則(或說增長出額外分支)取代為中止符時。

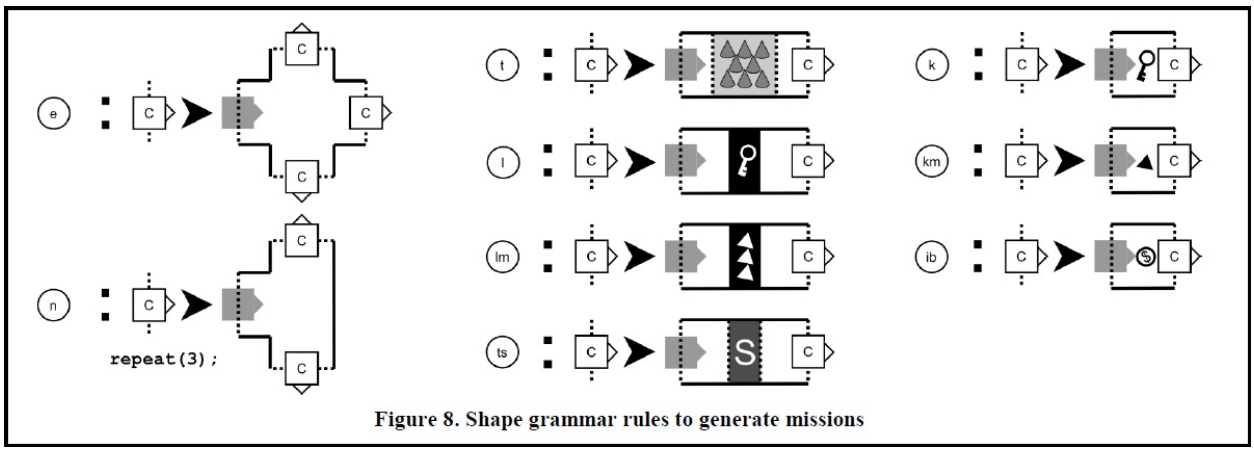

Figure 8. Shape grammar rules to generate missions

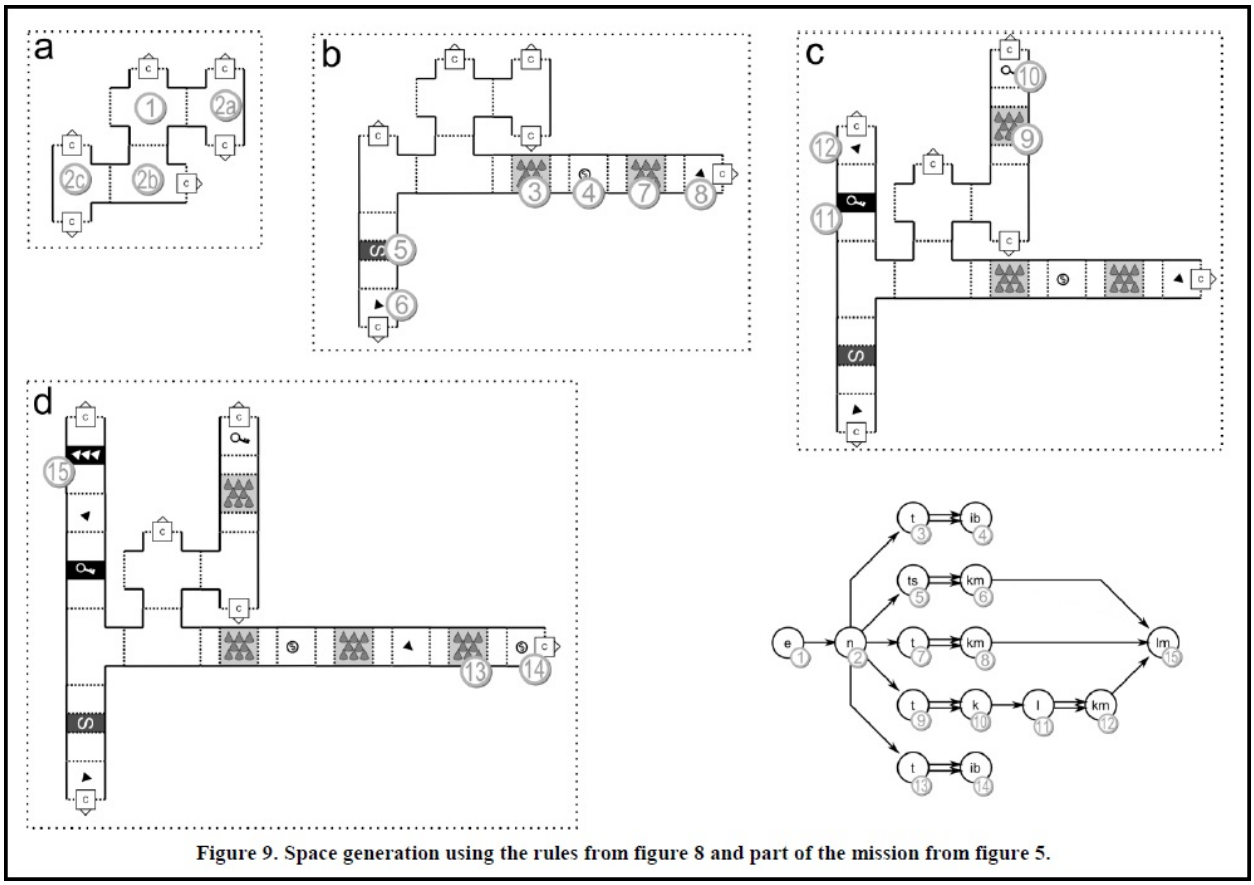

Figure 9. Space generation using the rules from figure 8 and part of the mission from figure 5.

圖 8列出形狀語法根據此方法而產生出的一些規則。圖 9展示出基於文上圖 5所示的任務第一部分的關卡結構的一些迭代。理論上要產生一個可以容納複數個任務並不會很困難。多個任務可以混合使用，透過生產器在多個任務中選擇下個被配置到地圖的工作。換而言之，第二個任務也可以在第一個任務預備後，被當作建造指令使用。

7. 涉及玩家的遊玩表現 (Involving Player Performance)

本論文討論之（部份的）生成技術也可以用於遊玩期間的關卡生成，玩家有機會以實際的遊玩表現來影響生成的結果。良好的策略是在關卡開始前能生成出一個任務，而且能確保關卡具趣味性的整體結構，同時空間也會響應玩家的遊玩歷程而成長。由於遊戲世界在遊玩的過程中生成，可能涉及了不同**空間規則 (Space rules)** 的動態權重，這使得玩家的實際遊玩表現可以被匯入至遊戲世界的構築之中。舉例來說，如果玩家已經遭遇許多怪物並與其對戰，則將降低「產生更多**怪物 (monsters)** 的規則」之權重，而提昇「產生不同類型**障礙 (obstacles)** 的規則」之權重，如此將確保遊玩的多樣性。或者是當玩家表現出她很享受這些戰鬥的過程（例如她會去追擊所有她所遇見的怪物），我們便可能讓她遭遇更多、更強大的怪物。在玩家實際的遊玩表現與遊戲生成之間所構成的**反饋環 (feedback loop)** 提供了許多契機。

較為簡單的方式是在已生成的空間中留下少量的**非終端節點 (non-terminals)**，以便在遊戲過程中被替換。像是非終端節點可以指定在特定的地下城房間中出現**障礙 (obstacle)** 或**獎勵 (reward)**，但沒有指定障礙物與獎勵的性質為何，直到玩家進入房間或開啟容器才會觸發非終端節點的替換動作，才決定出障礙物與獎勵的性質。這允許了遊戲能夠動態地改變難度與獎勵，以便反映玩家的遊玩表現與狀態。

另一方面，較為困難的情況是在遊玩的過程中生成任務。最好的策略是在構築空間之前，生成一個仍有一些非終端節點的任務，而替換非終端節點的動作應在遊玩過程中進行，並且應該根據玩家的遊玩表現直接或間接地通知。空間可以響應任務的變化而自動成長，抑或是已經適應了所有的可能性。**fractal story**由 Marie-Laure Ryan 提出的互動式結構相當淺顯易懂，當玩家被故事中某些劇情給吸引住後，故事便持續地生成更詳盡的內容 [20]。

8. 結論

動作冒險類遊戲的關卡是由空間與任務所構成的雙重結構。當程序化生成這類型的關卡時，最好能夠把生成過程拆分成兩個步驟。**圖狀語法 (Generative graph grammars)** 適合生成任務，它們能夠產生非線性結構優於線性結構，像是探索類的遊戲。同時它們能夠抓住一個良好遊戲體驗所需的大型架構。一旦任務被生成，**形狀語法 (Shape grammars)** 的擴展形式便可以增長出一個能容納生成任務的空間。這需要對形狀語法的通俗做法做一些修改，最重要的修改內容是將**形狀語法的規則**與**語法中用於生成任務的終端符號**建立關聯。

將過程拆分成兩個步驟，可以讓我們從每一種類型的語法中取得優勢。伴隨著一套別出心裁的規則與巧妙地使用遞迴，能夠用來生成有趣且多樣的關卡，而這些關卡不僅在探索時相當有趣，亦提供了完整的遊戲體驗。此外，這些技術能夠在遊玩的過程中生成關卡，讓遊戲世界能夠依照玩家的遊玩表現做響應。這將開拓了遊戲與**互動式故事 (interactive storytelling)** 交融的契機，在過去這是沒有被考究過的。

雖然程序化生成的原理背後是針對特定遊戲的實作，但**語法 (grammars)** 本身並不是。任務與空間語法必須建立在清晰、且能夠透徹遊戲最終樣貌的視角之上。此外，語法的品質將會是影響遊戲品質的關鍵要素，語法的建立需要專家級遊戲設計師的涉入或是使用這裡未提及之**演化式演算法 (Evolutionary algorithms)**。然而，利用**任務與空間語法 (Mission and Space grammars)** 去生成高品質的動作冒險類遊戲的關卡是相當有效的。