National Taiwan University of Science and Technology
Department of Computer Science and Information Engineering

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

Ze-Hao Wang

Master Thesis Oral Defense

July 27, 2017

Prof. Wen-Kai Tai	Prof. Kai-Lung Hua	Prof. Tung-Ju Hsieh	Prof. Wen-Huang Cheng
Committee Chairperson	Committee Member	Committee Member	Committee Member

Agenda

- Introduction
- Related Works
  - Mission / Space framework
  - Map Sketches & Evolution of Segments
- Proposed Methodologies
  - System Framework
  - Mission Grammars
  - Room Definitions and Instruction
  - Genetic Algorithm in Segments Evolution
- Experimental Results
- Conclusions and Contributions
- Future work

Introduction

Motivation

- [會再翻成英文]
- 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。我們預期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。
- 複雜系統

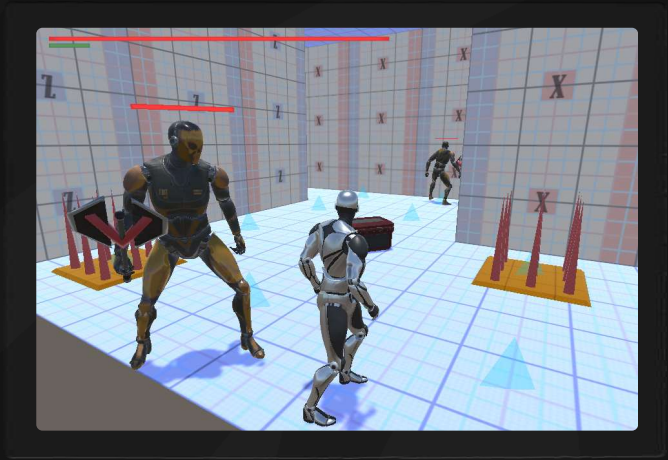
Dungeon Architect

DunGen

	Dungeon Architect	DunGen
Build topology	Randomly PCG, Manually	Randomly PCG
Mission-based topology	None	Stable-linear structure

Research Goals

- [會再翻成英文]
- 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。



Our Method

	Our Method	Dungeon Architect	DunGen
Build topology	PCG	Randomly PCG, Manually	Randomly PCG
Mission-based topology	Dynamic-nonlinear, meaningful structure	None	Stable-linear structure
Game Patterns	Genetic Algorithm	None	None

Related Works

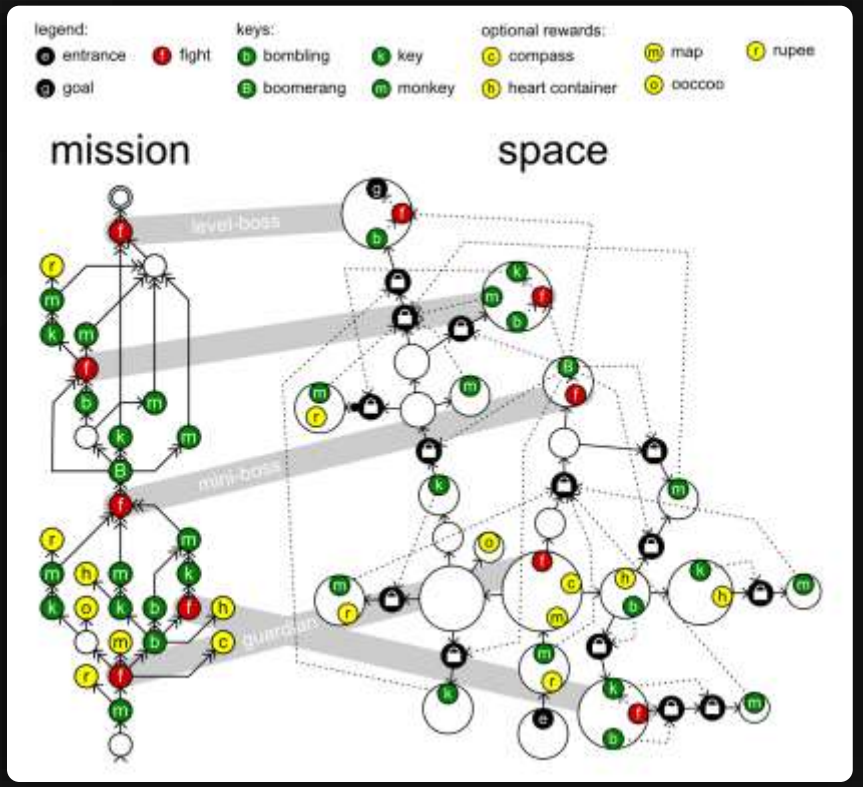
Mission / Space framework

Map Sketches & Evolution of Segments

Mission / Space framework

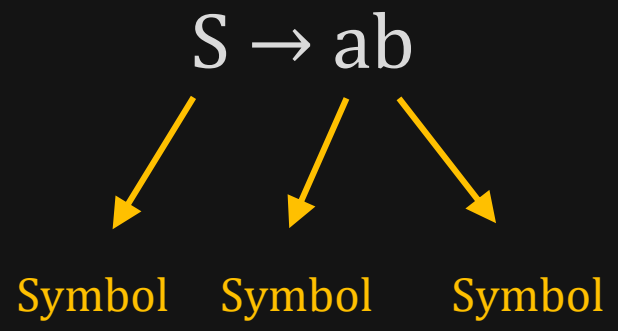
Mission/Space framework, focuses on **level design** and the **mechanics** that control player progression through a game.

- Transformational Grammars
- Mission Grammars
- Space Grammars
- Mission Graph Convert into Space Graph



Transformational Grammars

- Be consisted from an alphabet and a set of rules
  - **Alphabet**
    It is a set of symbols the grammar works with.
  - **Set of rules**
    It specifies what symbol can be replaced by what other symbols to form a new string.
- Sides and symbols of the rule
  - **Terminals** (common in lowercase)
    Symbols in the alphabet can never be replaced because there are no rules.
  - **Non-terminals** (common in uppercase)
    Symbols have rules that specify their replacement.



{ S, a, b } is an alphabet

Mission Grammars

-  **Inhibitions**

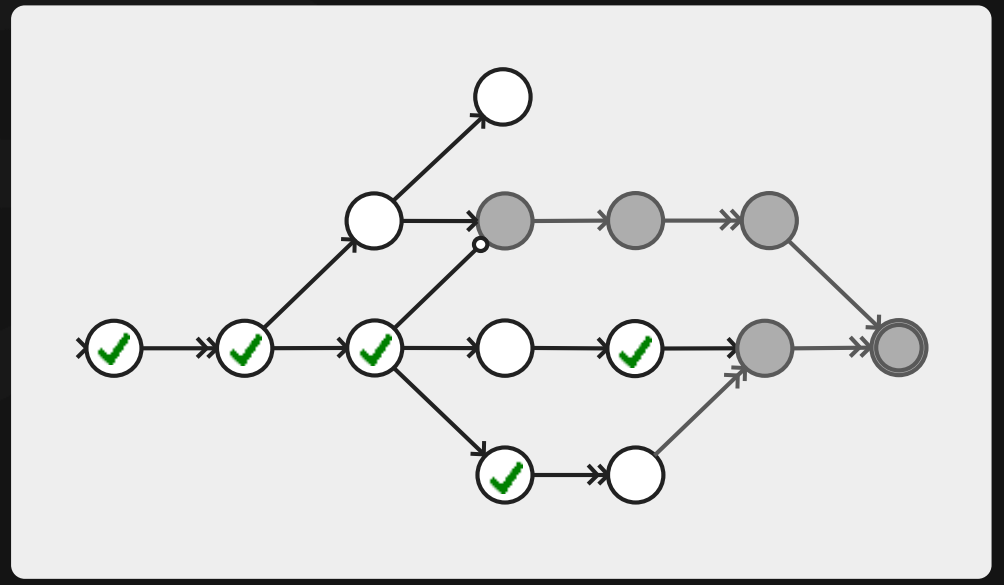
  Available when at least one of the weak prerequisites is completed.

-  **Strong requirement**

  Available when all strong prerequisites are completed.

-  **Weak requirement**

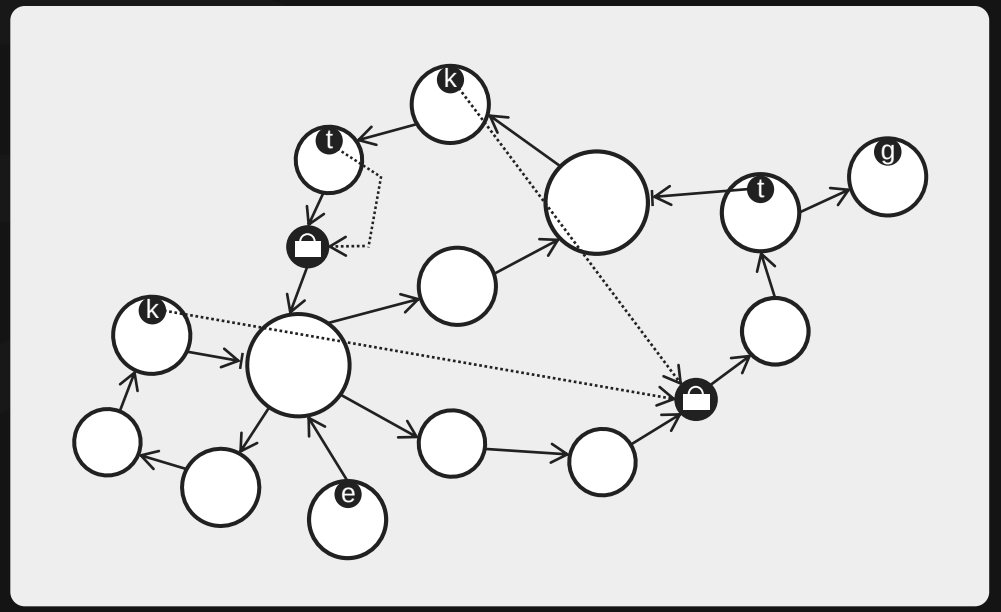
  Available when at least one of the weak prerequisites is completed.



Example of mission graph

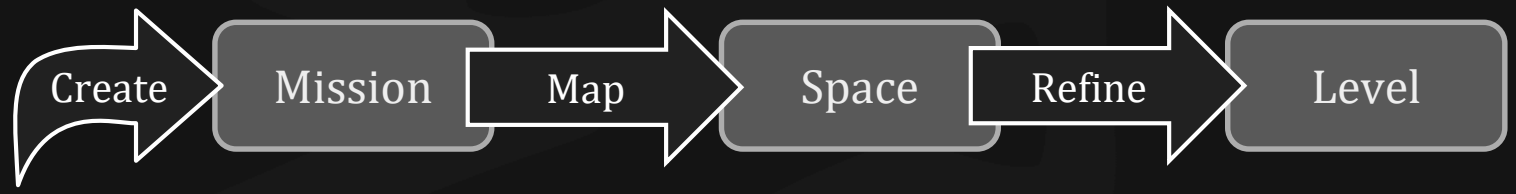
- Mission graphs represent the players' progress towards a goal not by tracking their physical location, but by tracking the tasks they must perform to finish a level.
- A mission graph is a directed graph that represents a sort of to-do list with each node representing a task that might or must be executed by the players.

Space Grammars

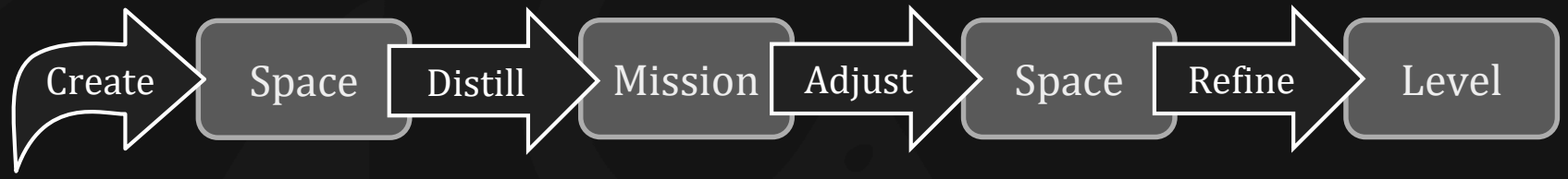


Mission Graph Convert into Space Graph

- The steps in the generation progress investigated in this paper in detail:



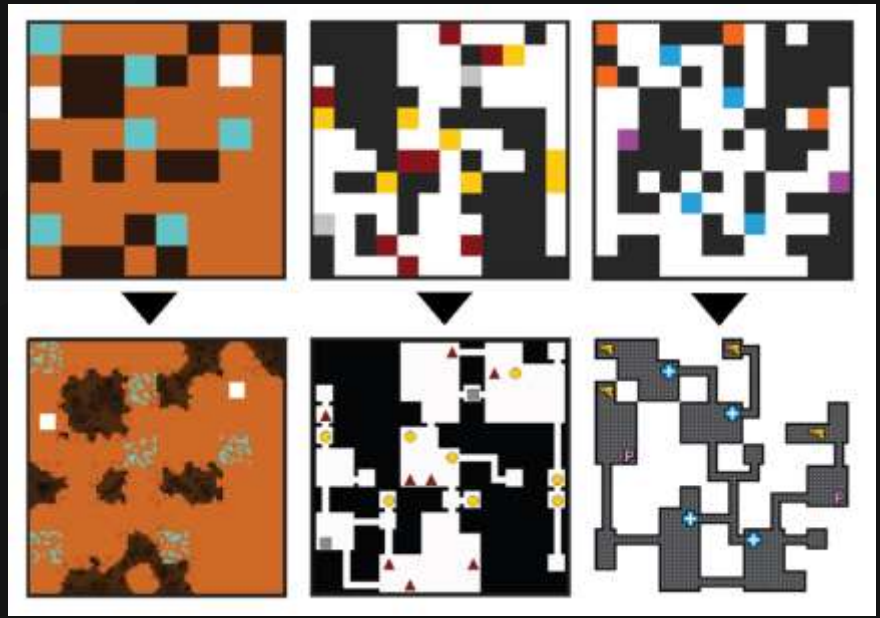
- An alternative method, does not go into detail:



Map Sketches & Evolution of Segments

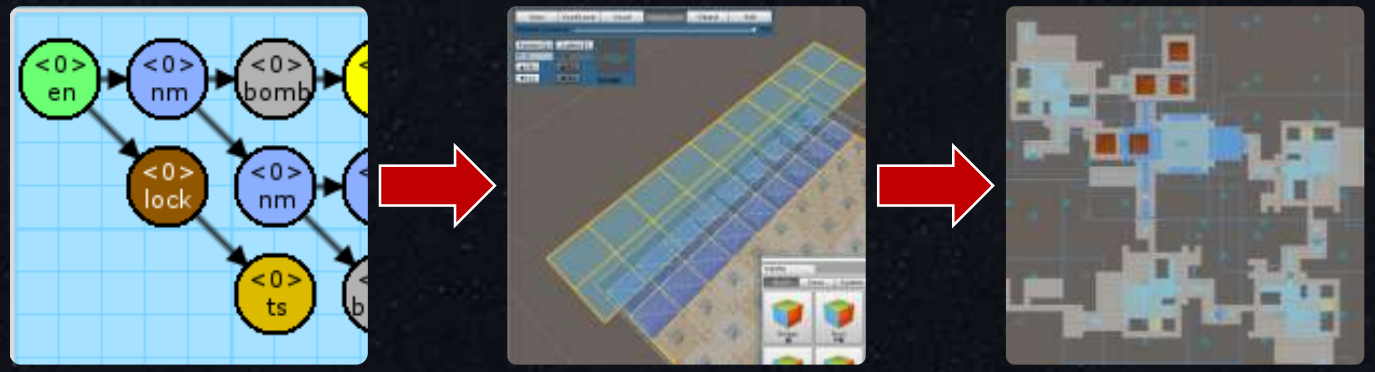
Under construction.

- Map Sketches
- Map Sketch Evolution
- Dungeon Segments
- Dungeon Segment Evolution



Proposed Methodologies

System Framework



- Phase 1 -

Mission / Space framework

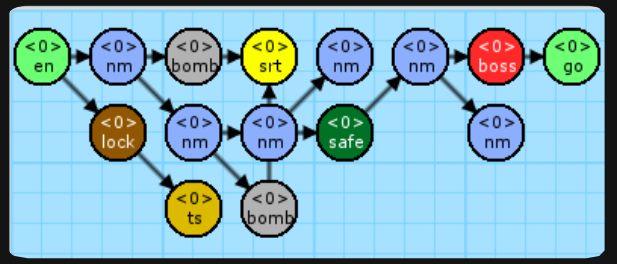
- Phase 2 -

Dungeon Segment Evolution

System Framework

1. Create Mission Grammars

- Mission Alphabet
- Mission Rules
- Mission Graph
  - Rewrite System I
  - Random seed

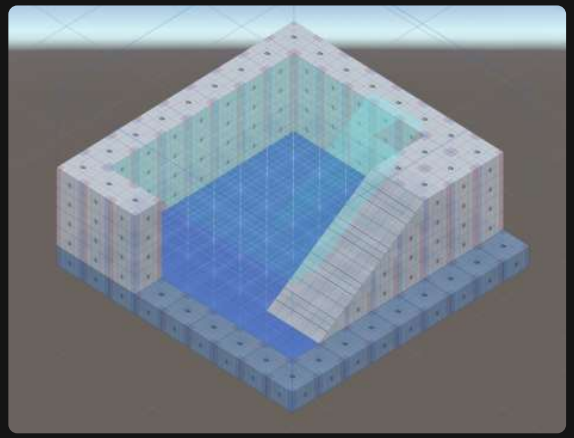


Example of Mission Graph

System Framework

2. Create Volumes

- Voxel-based units
- Decorations
- Connections
  - Entrance
  - Exit

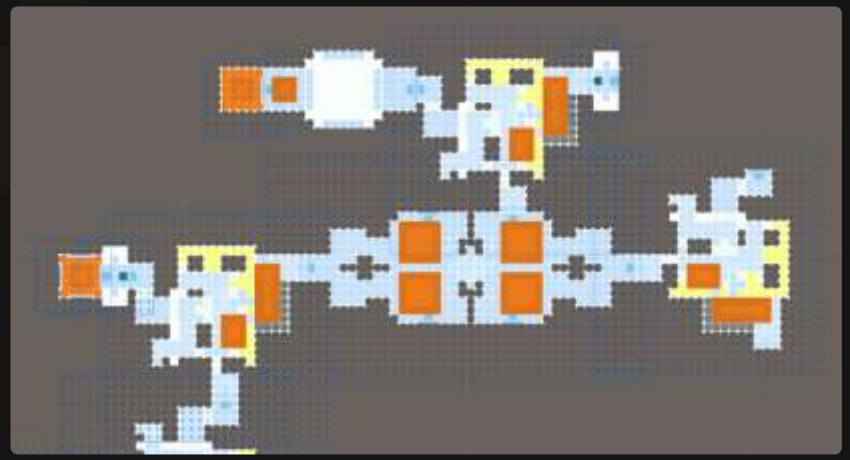


Example of volume

System Framework

3. Generate the Space

- Rewrite System II
  - Instruction
  - Replacement
  - Collision issue

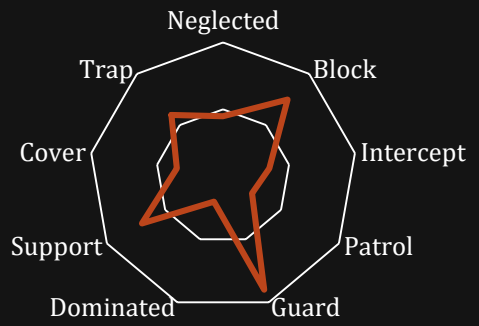


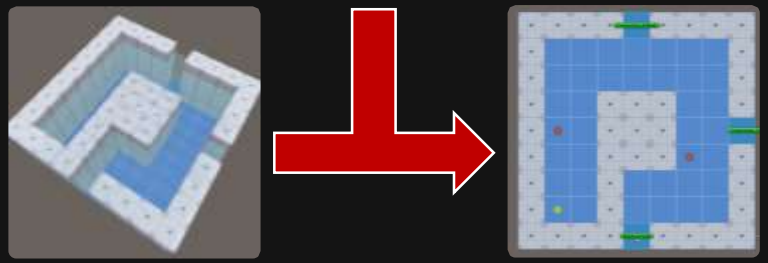
Example of space

System Framework

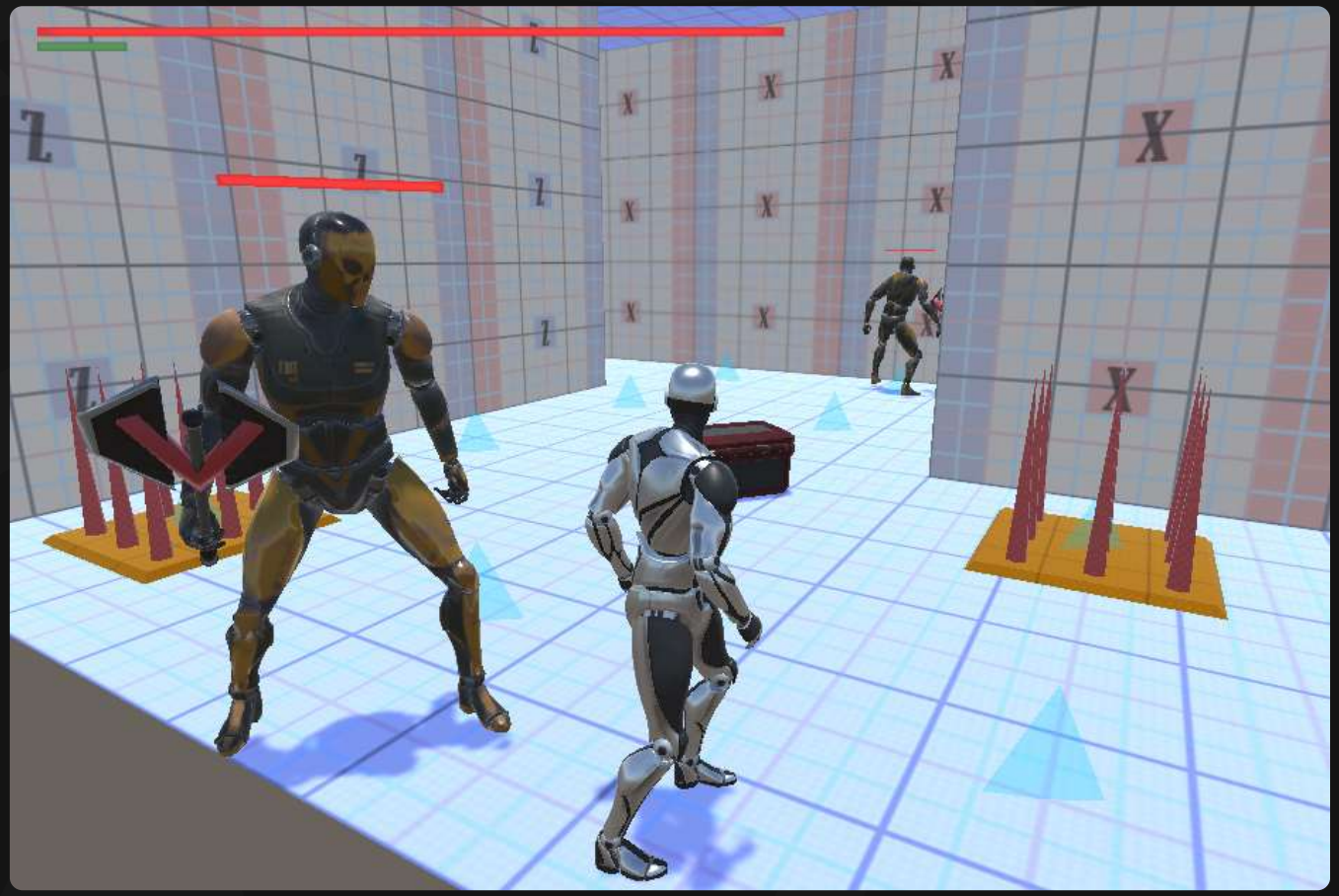
4. Generate the Gameplay Patterns

- Genetic Algorithm for emergence objects
- Nine main metrics to design the fitness functions

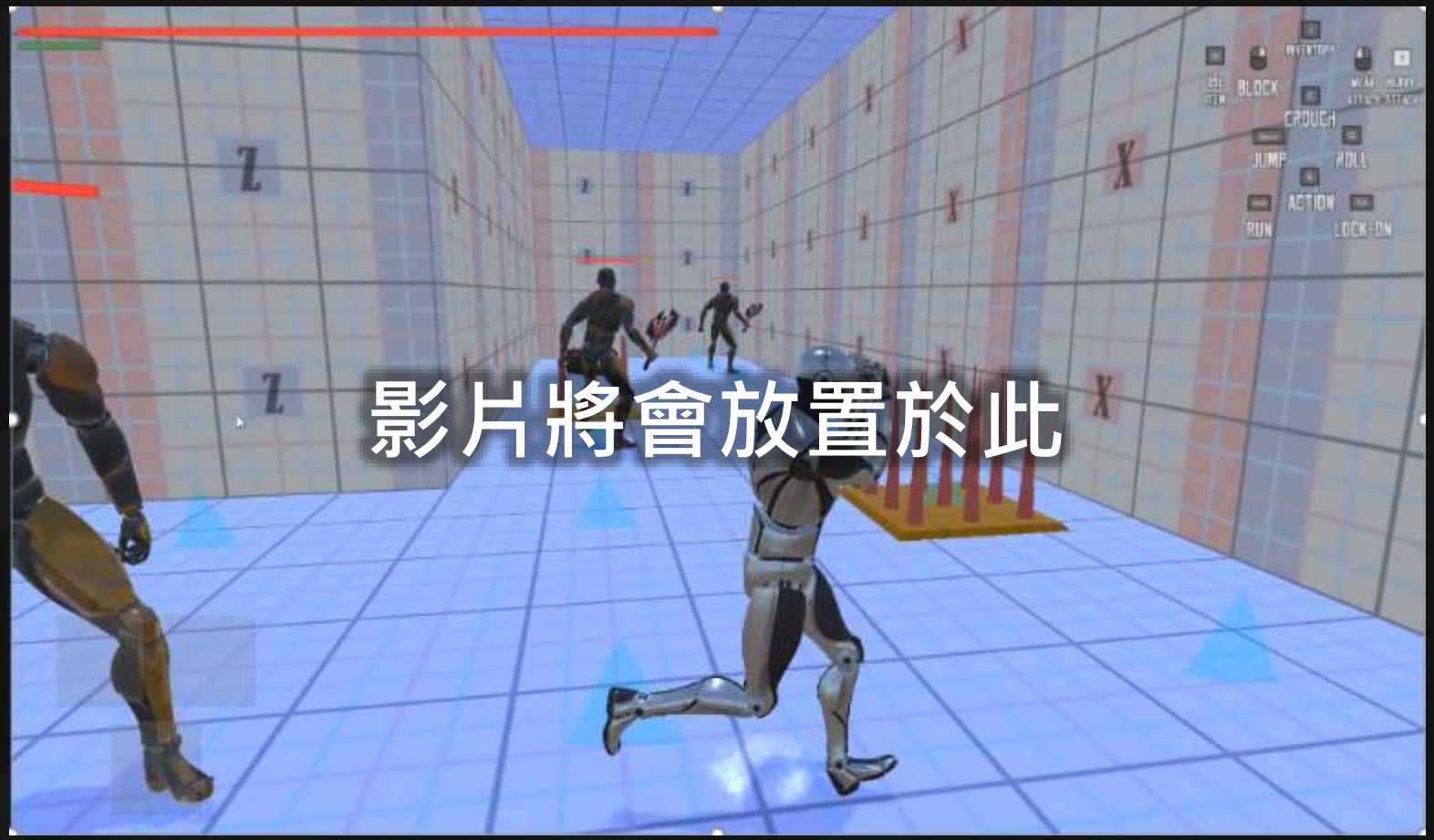




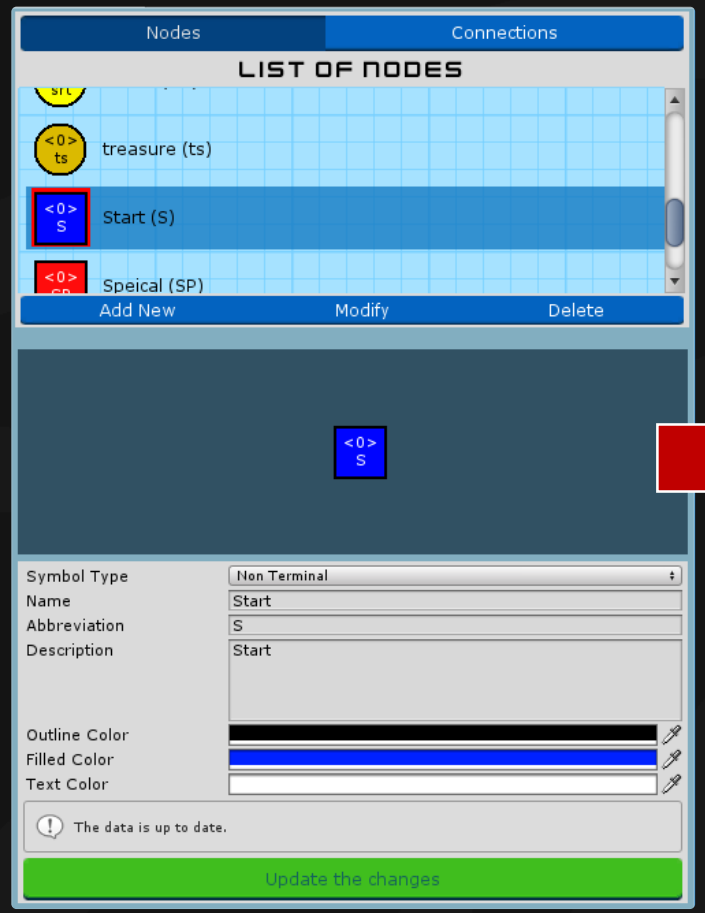
System Framework



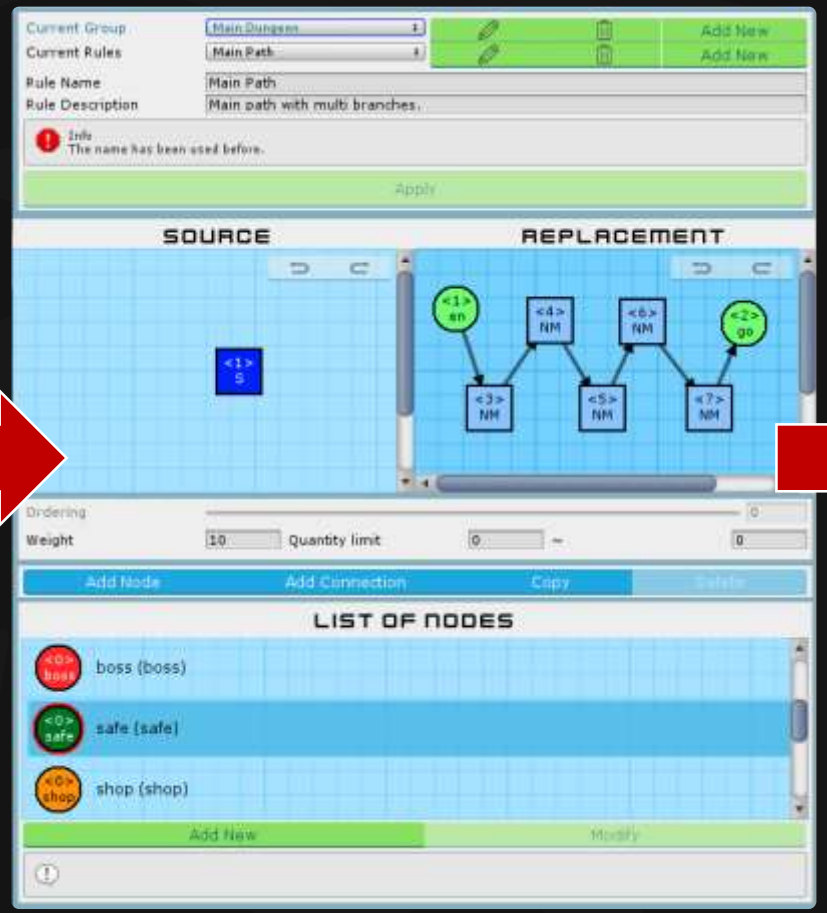
System DEMO



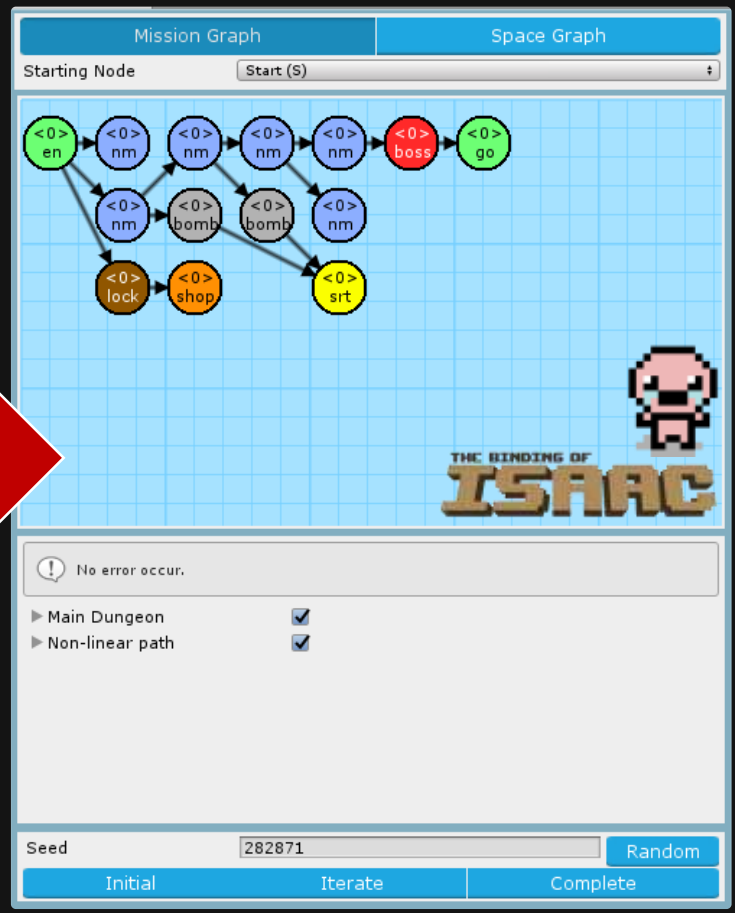
Mission Grammars



Mission Alphabet

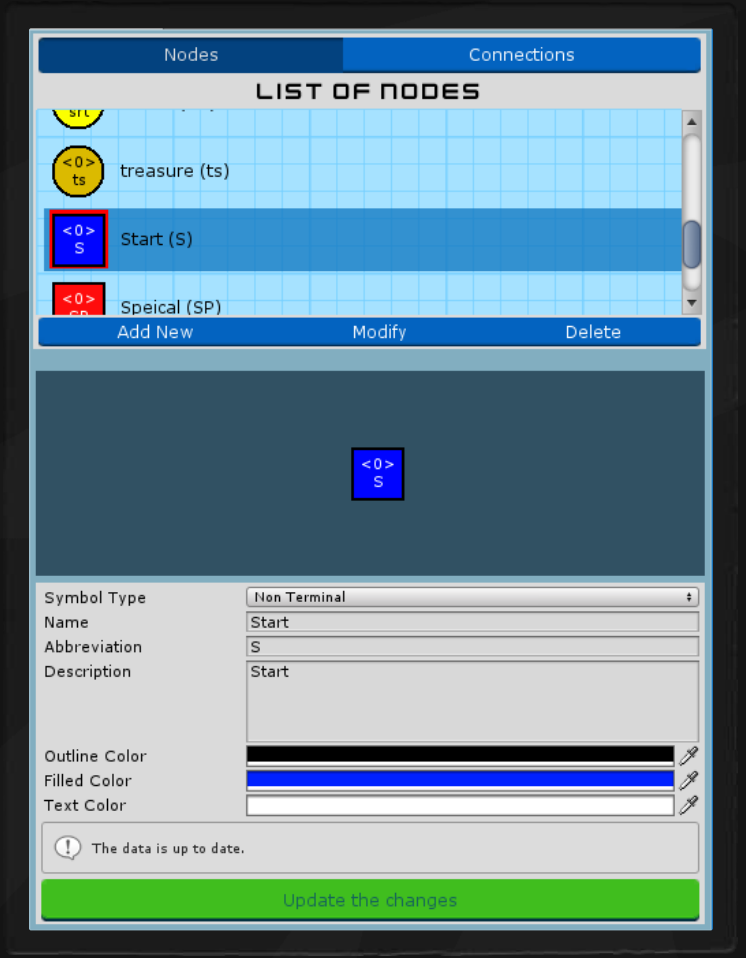


Mission Rules



Mission Graph

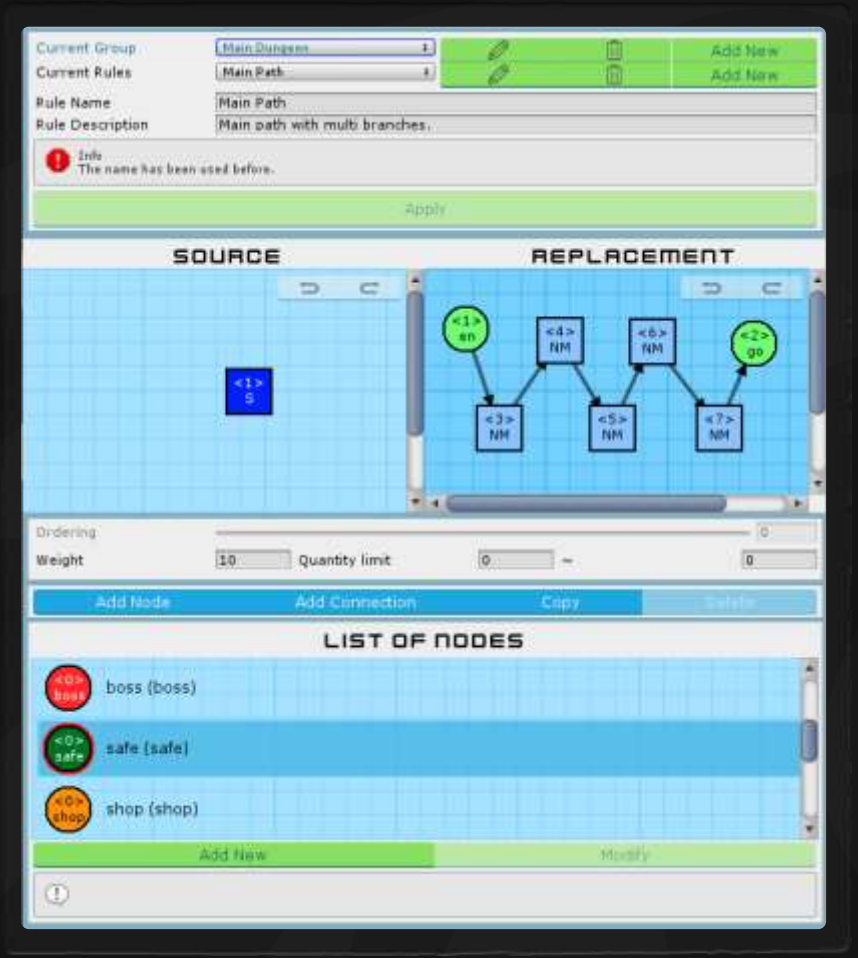
Introduction of Mission Alphabet



Mission Alphabet Window

- List of symbols
  - Nodes and connections
  - Directly preview before selected
- Preview area
  - Preview the symbol after editing immediately
  - Submit hint and form validations
- Default and extended nodes
  - Default: *None*, *Entrance*, *Goal*
  - Extended: *Any*

Introduction of Mission Rules



Mission Rules Window

- Hierarchy structure
  - Groups > Rules > Graph Grammar of Left-hand side & Right-hand side Rules
  - Friendly interface to create, delete and edit.
- Rule canvas
  - Drag & drop to modify symbols
  - Custom size of canvas
  - Selected symbol highlighting
  - Connections stick on nodes automatically
  - Back trace the states (Redo / Undo)

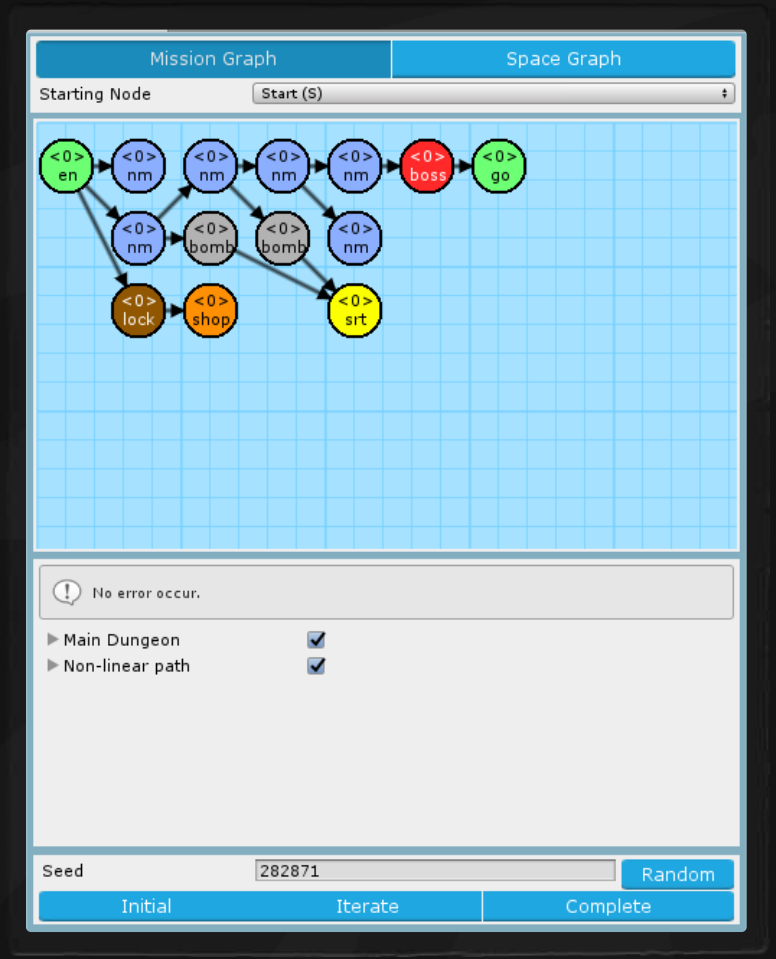
Avoid the illegal mission rules

- <放非法規則的圖 !!>

Avoid the illegal mission rules

Illegal pattern	Description
Left More Than Right	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。
Empty Left	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。
Isolated Node	孤立的節點將會導致任務圖破碎。
Isolated Connection	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。
Exactly Duplicated	若左右規則同構將導致改寫系統陷入無限循環。
Multiple Relations	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。
Cyclic Link	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。
Orphan Node	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。
Overflowed Any Node	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。

Introduction of Mission Graph



Mission Graph Window

- Set starting node
  - The head of the mission graph
- Preview mission graph
- Rewrite system I
  - Based on the selected rules in mission grammars
  - Find the matched subgraphs using rules
    - VF Graph Isomorphism Algorithm [1]

[1] VF Graph Isomorphism Algorithm: https://www.codeproject.com/Articles/23144/A-C-Implementation-of-the-VF-Graph-Isomorphism-Alg

Rewrite System I

Pseudocode of Rewrite System I

```
1  Algorithm RewriteSystem1 (Node root)
2      if matchedRule is found from root: // If not found then skip
3          matchedRule = FindMatchs(TransToVFGraph(root)) // Transfer root into VF Graph
4          Remove the connections of matched parts with matchedRule
5          Replace the nodes based on right-hand side
6          Append the nodes that additional index in right-hand side
7          Re-add the connections based on right-hand side
8          Remove the index information of nodes in graph
9      for child := root->children: // DFS
10         RewriteSystem1(child)
11     return root
```

FindMatchs with the Random seed

```
1  Function FindMatchs (VFGraph vfGraph)
2      for rule := rules.shuffle(seed) // Random shuffle via random seed from random table
3          if rule.QuantityLimitMax == 0: // If not found then skip
4              continue
5          if rule.leftHand is isomorphic to vfGraph: // Means the subgraph of vfGraph
6              rule.QuantityLimitMax -= rule.QuantityLimitMax > 0 ? 1 : 0;
7              return rule
8      return root
```

- Random table
- Original “0” value has transferred to “-1” means the rule is unlimited usage

Manual

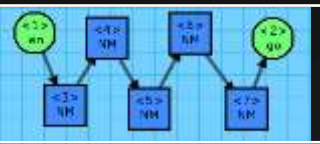
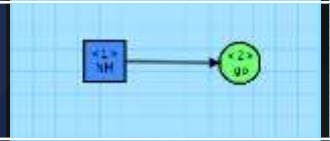
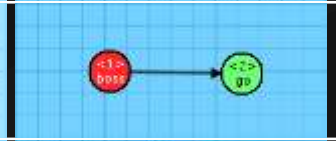
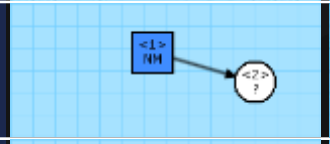
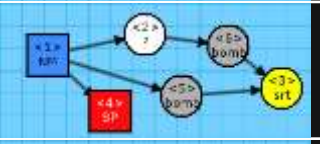
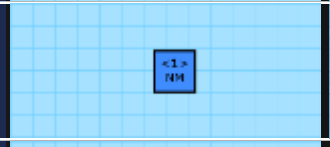
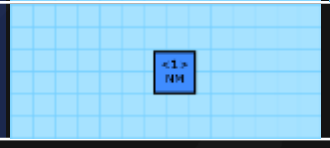
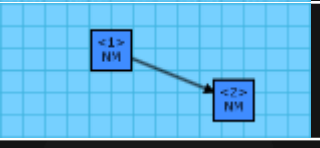
A Create the Mission Alphabet

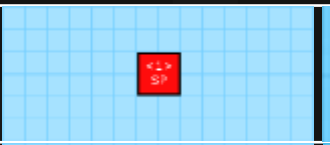
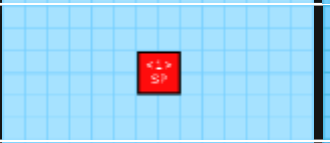
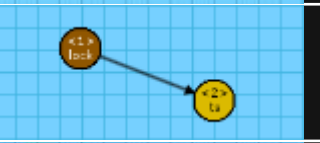
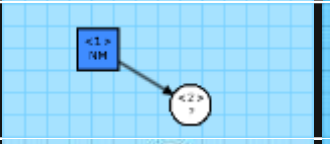
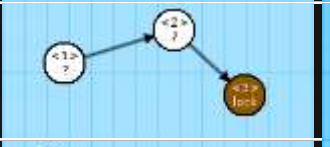
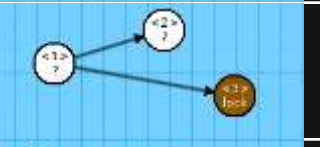
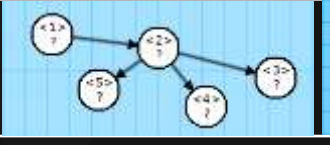
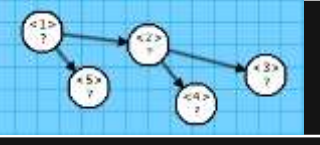
Node	Start (S)	entrance (en)	Normal (NM)	normal (nm)	boss (boss)	goal (go)	safe (safe)	Special (SP)	shop (shop)	bomb (bomb)	lock (lock)	secret (srt)	treasure (ts)
Display													
Quantity limit	0	0	0	8	1	0	0	0	1	0	0	1	1
Require-ment					Battle and end			Multi - choice	Get the key from secret, then enter the secret or treasure room behind the wall that need to use the bomb to destroy it				

※ Limit of *Zero* Means infinite.

Manual

B Design the Mission Rules

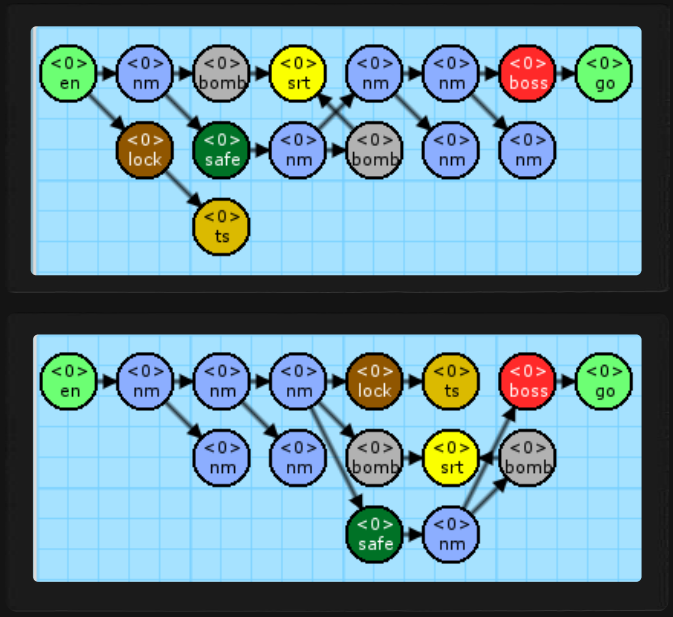
	Source	Replacement	
Main Path			Weight: 10 Limit: [0, ∞]
Boss Room			Weight: 150 Limit: [0, ∞]
Set Secret			Weight: 30 Limit: [1, 1]
Expora-tion			Weight: 10 Limit: [0, ∞]
More Branch			Weight: 10 Limit: [0, ∞]

	Source	Replacement	
Shop			Weight: 10 Limit: [0, 1]
Treasure			Weight: 10 Limit: [0, 1]
Extend Safe			Weight: 10 Limit: [0, 1]
Forward Lock			Weight: 10 Limit: [0, ∞]
Flat Rooms			Weight: 10 Limit: [0, ∞]

Manual

C Export the Mission Graph

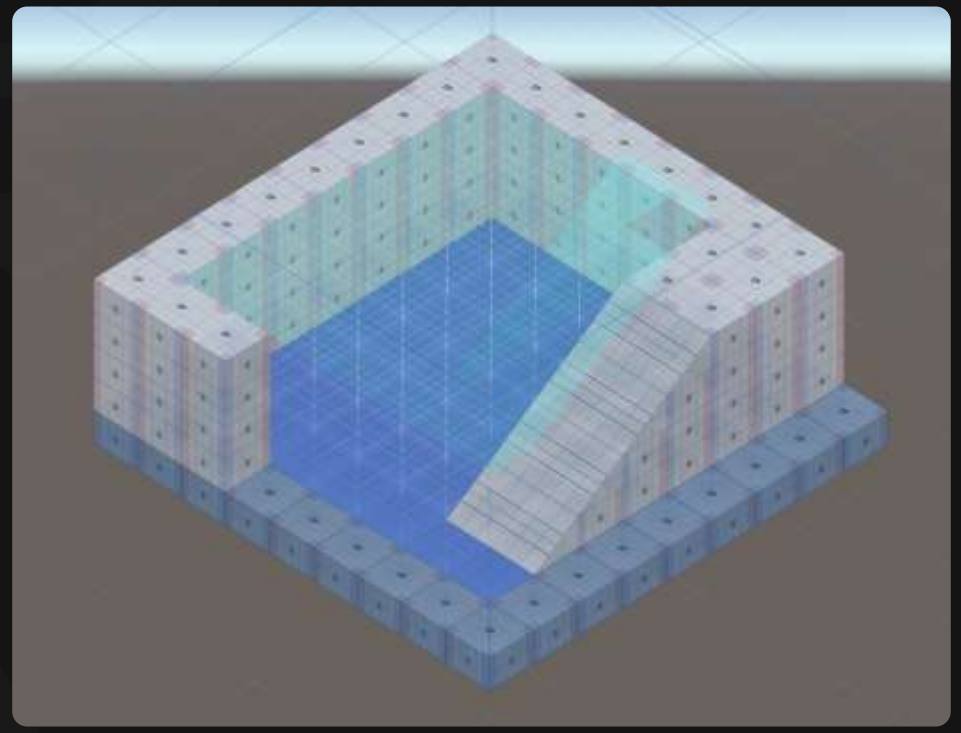
- Set the starting node
  - Start to execute the iteration from the root.
- Set the random seed (Random number table)
  - Seed affects selection when there are many candidates, or and the additional iterations [1] in the moment.



[1] additional iterations: The completed generation means all nodes in the graph are replaced to terminal nodes. Additionally executes more iterations to parallel rewrite the current graph.

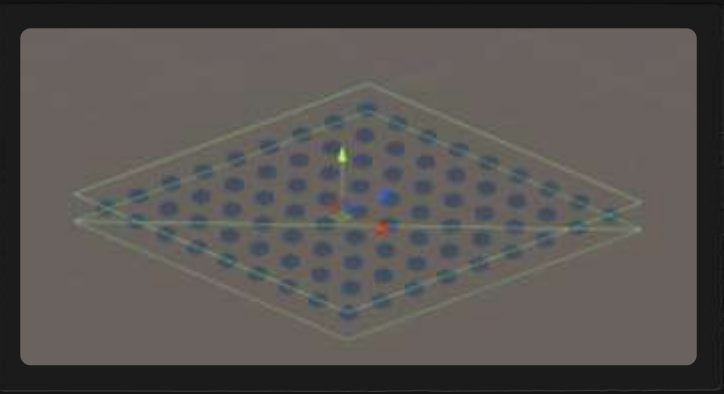
Room Definitions and Instruction

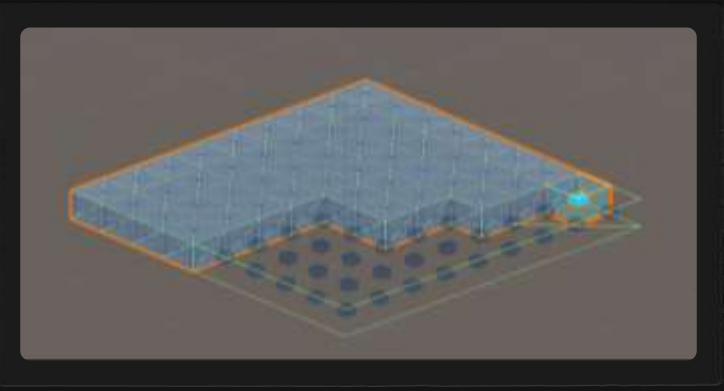
- Editor
  - Voxel-based units
  - Markers I - Decorations
  - Markers II - Connections
- Rewrite System II
  - Instruction
  - Replacement
  - Pseudocode



Voxel-based units in Editor

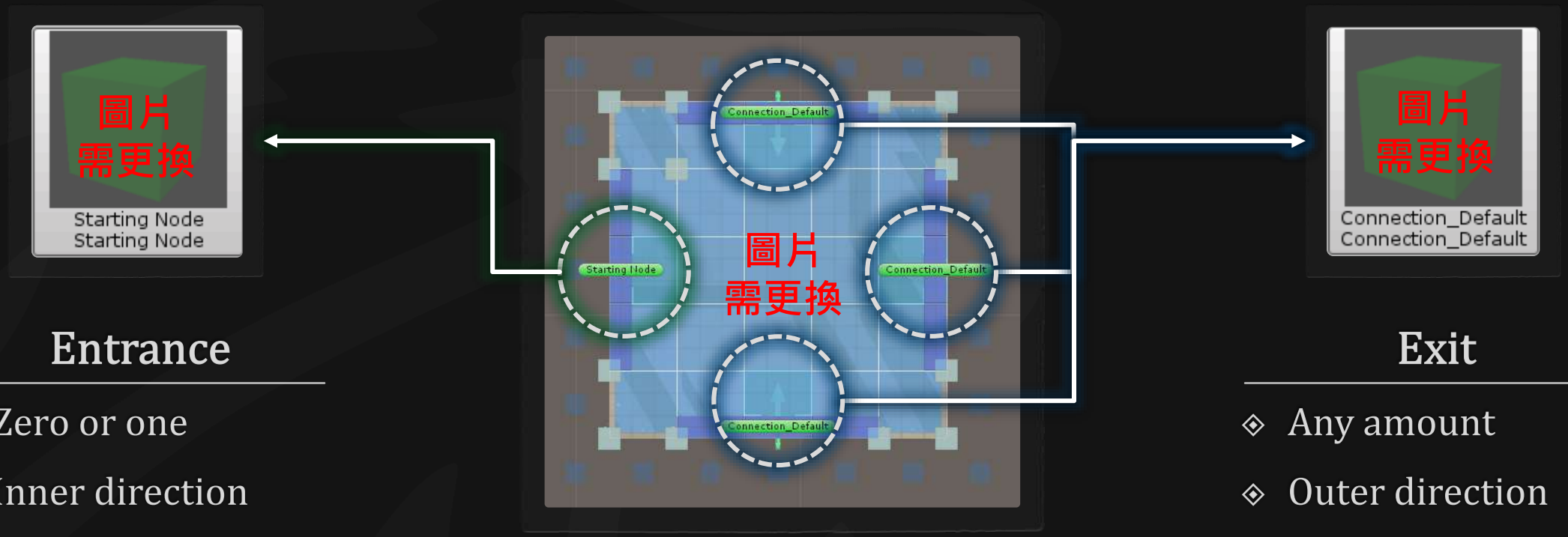
- Hierarchy data structure
  - **Level**
    A set of volumes. Expresses a complete game level.
  - **Volume**
    A set of chunks. Mostly expresses a room.
  - **Chunk**
    Consists of 9x9x9 voxel. Based on the size of volume, uses different numbers of chunk.
  - **Voxel**
    The minimum unit of level, size is 3x2x3. It's property is a block or decoration with nine orientations.





Decorations in Editor

Connections in Editor



Entrance

- Zero or one
- Inner direction

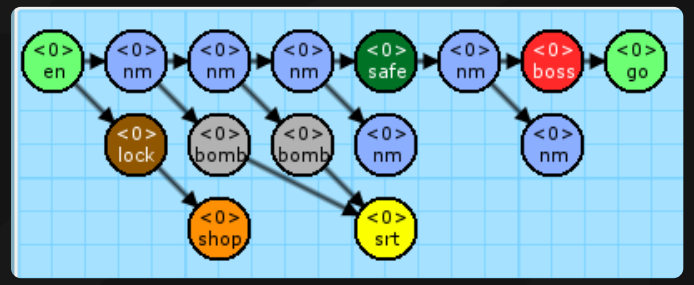
Exit

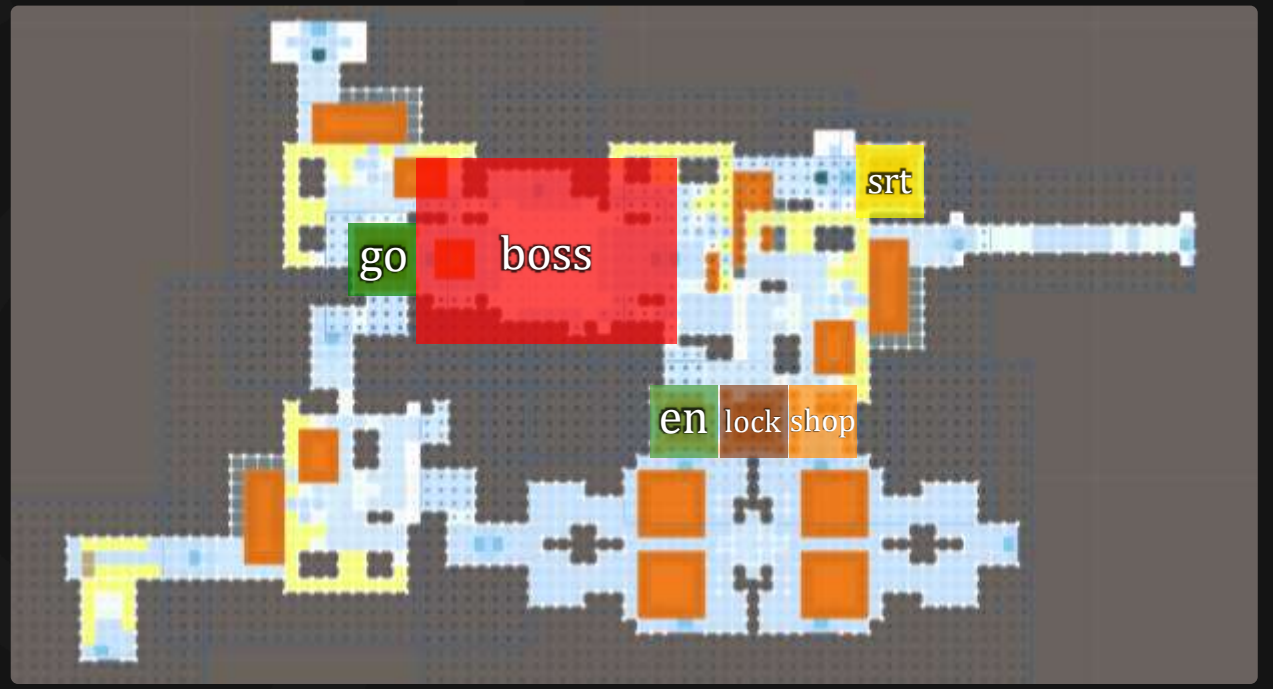
- Any amount
- Outer direction

- The **exits** of "**volume A**" will stick with the **entrance** of "**volume B**", according to their direction of arrow. If doesn't exist any entrance, will pick one exit randomly.

Rewrite System II

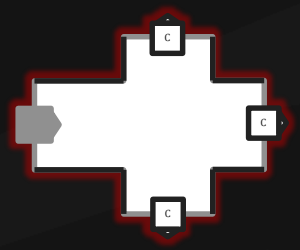
- Another rewrite system of Mission/Space framework.
- Generate the **Space** from **Mission Grammars**.



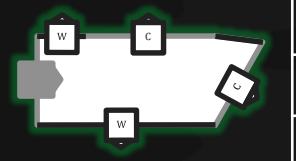


Instruction of Rewrite System II

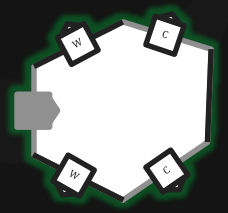
Space rules (simplify, expressed in 2D)



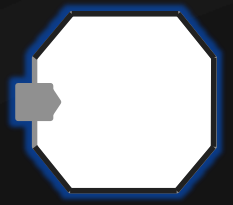
Rule	build entrance
Inst.	en (entrance)
Rep.	Entance



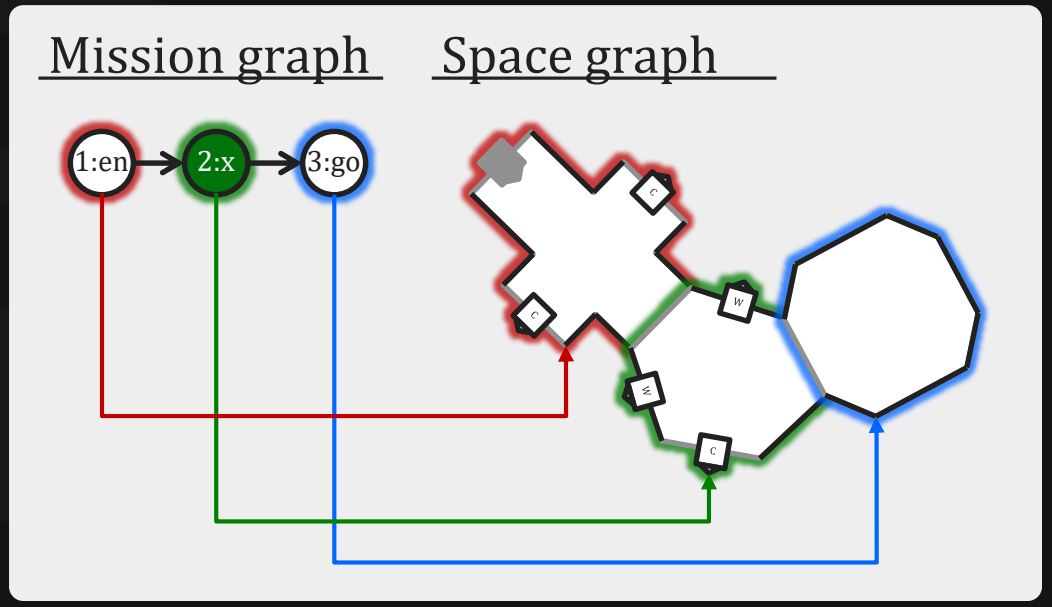
Rule	grow section 2
Inst.	x (explore)
Rep.	Connection



Rule	grow section 1
Inst.	x (explore)
Rep.	Connection



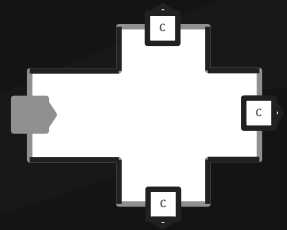
Rule	goal
Inst.	go (goal)
Rep.	Connection



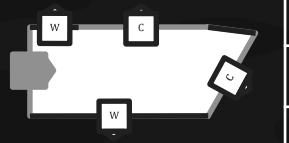
- Each rule in the shape grammar was associated with a **terminal symbol** in the mission grammar.
- Look for rules that implement that symbol, selects one at random based on their **relative weight**.

Replacement of Rewrite System II

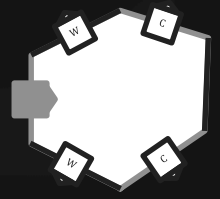
Space rules (simplify, expressed in 2D)



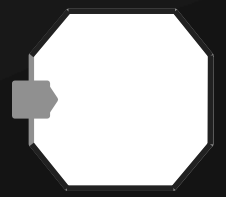
Rule	build entrance
Inst.	en (entrance)
Rep.	Entance



Rule	grow section 2
Inst.	x (explore)
Rep.	Connection



Rule	grow section 1
Inst.	x (explore)
Rep.	Connection



Rule	goal
Inst.	go (goal)
Rep.	Connection

Rule	close connection
Inst.	(none)
Rep.	Connection

Rule	protowall1
Inst.	(none)
Rep.	ProtoWall

Rule	protowall2
Inst.	(none)
Rep.	ProtoWall

Space graph

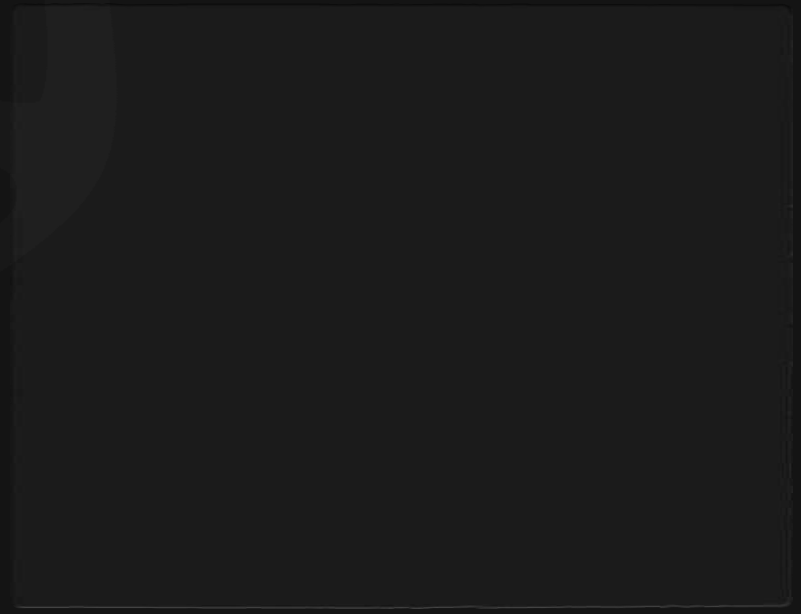
- After all nodes in mission graph having transferred into space, the **remaining connections** have to fit the completed space.

Pseudocode of Rewrite System II

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

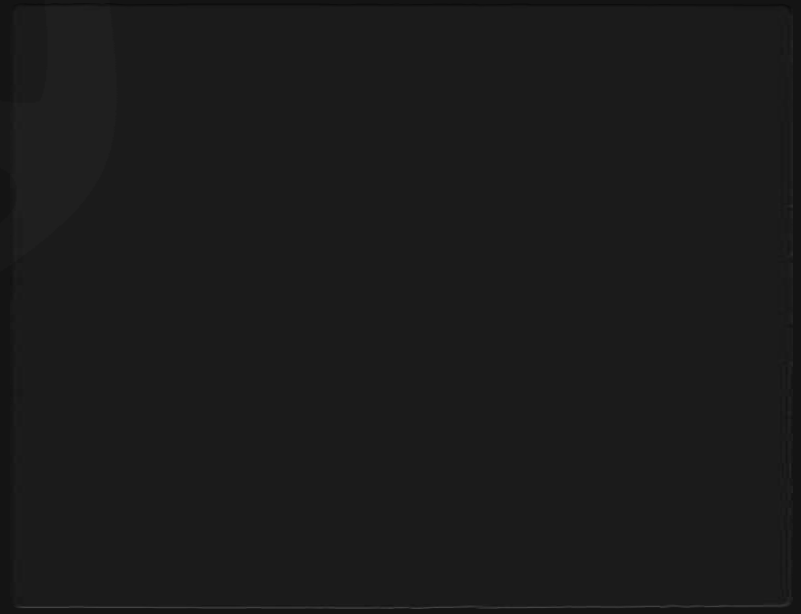
Manual

Design the Volumes



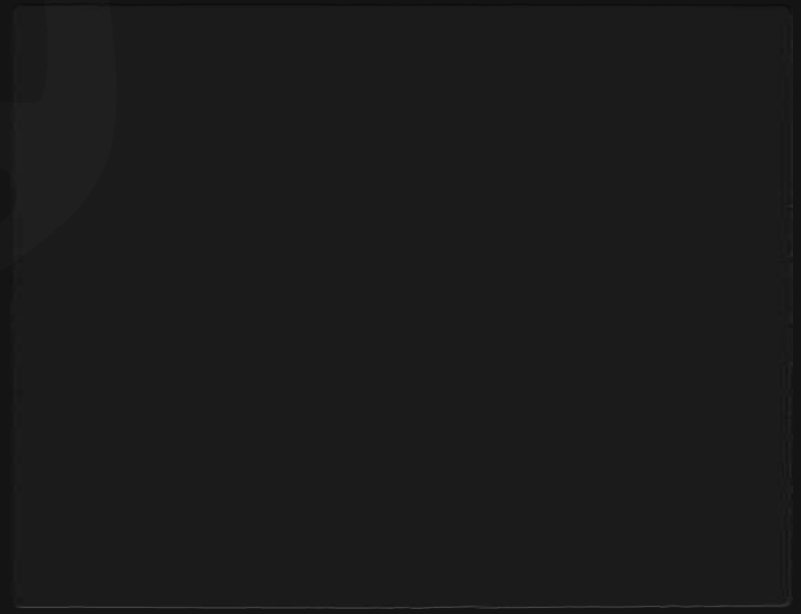
Manual

E Append the Starting Node & Connections



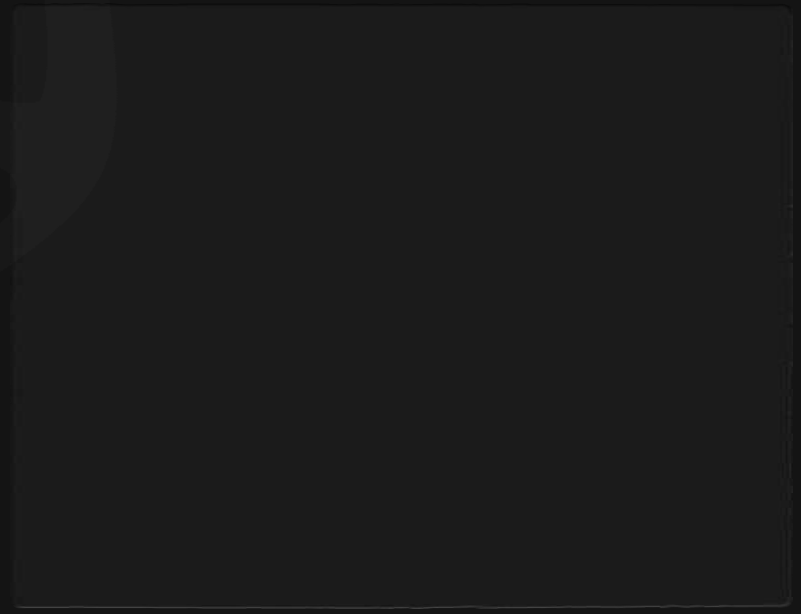
Manual

F Defined the Instruction of Rewrite System



Manual

- Defined the Replacement of Rewrite System



Genetic Algorithm in Segments Evolution

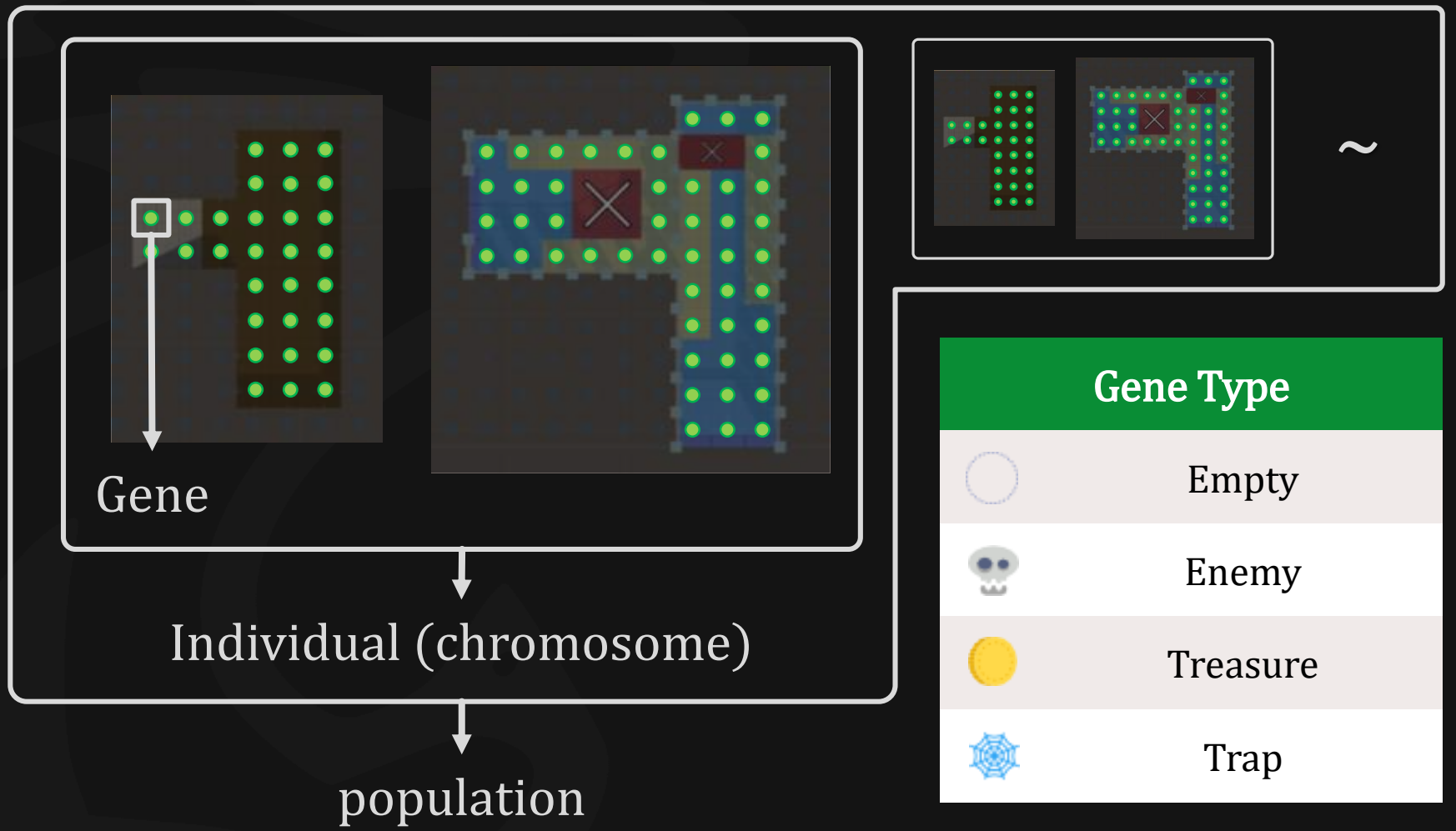
- Genetic Algorithm
  - Algorithm flow
  - Gene
  - Crossover
  - Mutation
- Fitness functions of GA
  - Common parts
  - Metrics

Algorithm flow

Gene

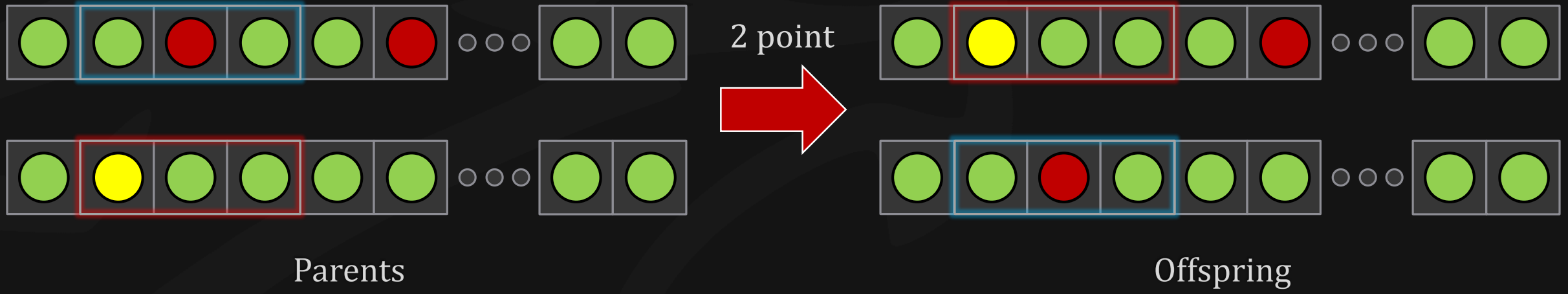
Basement

Ground



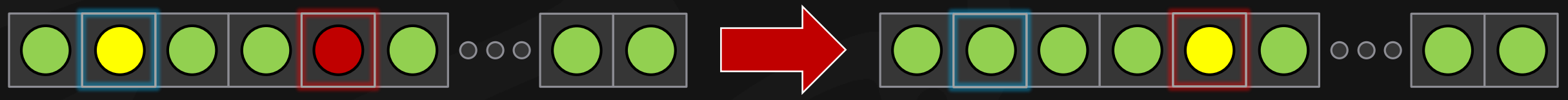
Gene Type
Empty
Enemy
Treasure
Trap

Crossover



- 80% crossover
- Two point crossover

Mutation



Parents

Offspring

- 10% mutation
- Change 5% ~ 20% positions each chromosome

Fitness functions of GA

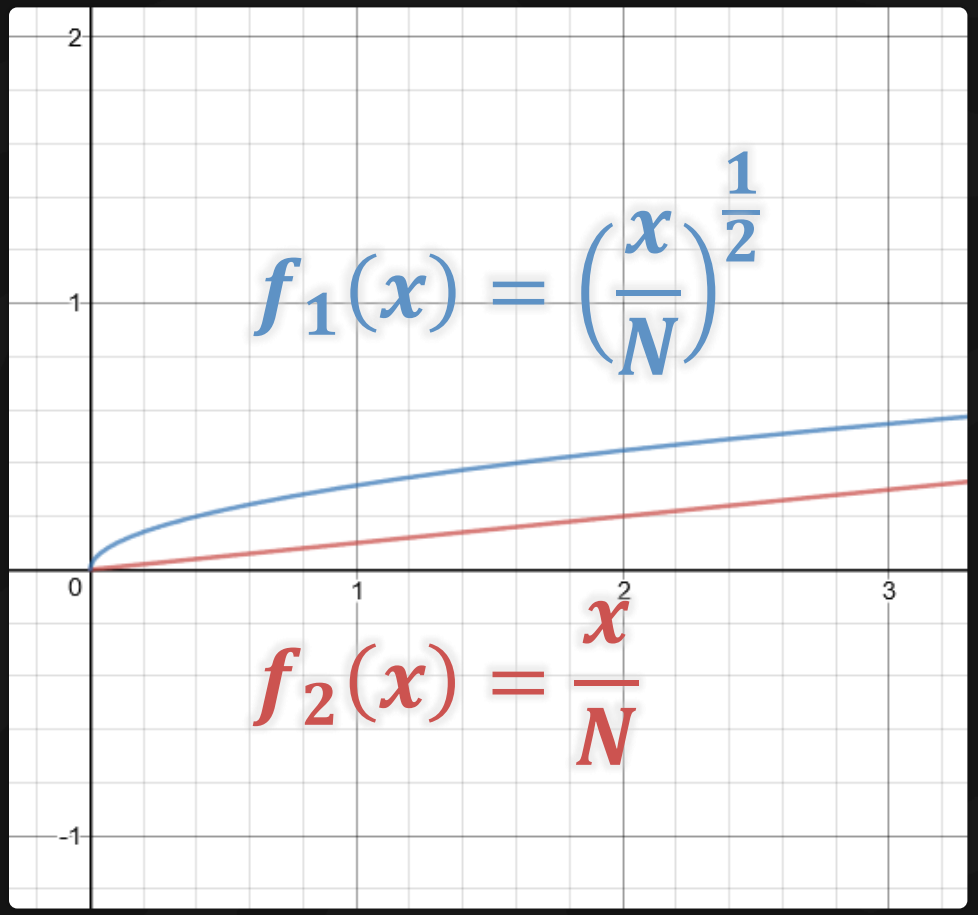
- Common parts
  - Main Path Calculation
  - Normalize the fitness scores
- Metrics
  - Neglected
  - Block
  - Intercept
  - Patrol
  - Guard
  - Dominated
  - Support
  - Cover
  - Trap

Main Path Calculation

- A-Star in three-dimensional

委託廣柏整理清單

Normalize the fitness scores

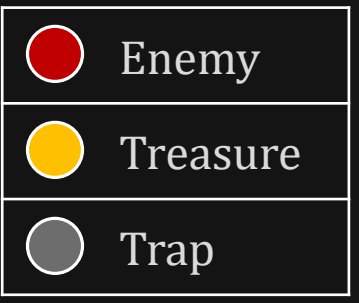


- Parameters
  - x : Un-normalized scored
  - N : Number of enemy
  - $f(x)$: Normalized scored
- An example. If $N = 10$
  - $f_1(0) = 0.0000$; $f_2(0) = 0.0000$
    +0.3162 +0.1000
  - $f_1(1) \cong 0.3162$; $f_2(1) = 0.1000$
    +0.1310 +0.1000
  - $f_1(2) \cong 0.4472$; $f_2(2) = 0.2000$
    +0.1005 + 0.1000
  - $f_1(3) \cong 0.5477$; $f_2(3) = 0.3000$

1st metric: Neglected

- Fitness function:

- Description:
  [會再翻成英文]
  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。



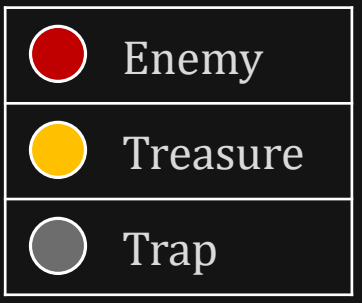
Example gameplay of "Neglected"

2nd metric: Block

- Fitness function:

$$f_{blk} = \log_{MP} E \, , MP = \sum_{i=1}^{N} mp_i \, , E = \sum_{j=1}^{M} e_j$$

- Description:
  [會再翻成英文]
  敵人會專注於阻擋玩家繼續前進，迫使玩家與其發生衝突。MP 為加總所有空間動線權重 mp_i；E 為加總所有敵人於空間之動線權重 (e_j)，倘敵人並未落在動線上，權重則為0。為了達到隨著動線上的敵人愈多，對此適應性函數的影響力愈低，則取以 MP 為底 E 的對數。

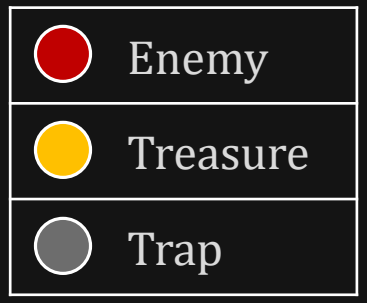


Example gameplay of "Block"

3rd metric: Intercept

- Fitness function:

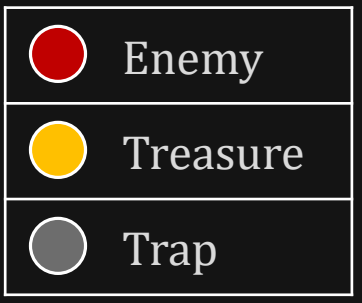
- Description:
  [會再翻成英文]
  與阻攔點近似，但敵人會被配置於動線附近非動線上，以快速追擊玩家為目的。各敵人 (E_i) 越接近空間動線各點 (MP_j) 時影響愈大，且動線權重 (mp_j) 亦會影響加權程度。



Example gameplay of "Intercept"

4th metric: Patrol

- Fitness function:



- Description:
  [會再翻成英文]
  為確保各敵人擁有足夠的空間能夠進行移動。利用 $Cover_i$ 計算敵人 (E_i) 在指定半徑 (r) 內，能夠行動的座標點數量比例，$count(E_i, r)$ 代表指定半徑內敵人可以行動的座標點數量； $plane(E_i, r)$ 代表以敵人為中心的指定半徑內，平面上所有座標數量（包含不可通行的牆壁等類型）。將 $count(E_i, r)$ 和 $plane(E_i, r)$ 的比例作為可以行動的座標數量比例。另外，本次實驗為三維空間，因此有機會出現可行走的數量大於平面上所有座標數量，在此對兩者比較大小，取最大值作為所有座標數量，以確保 $Cover_i$ 的數值介於0至1之間。

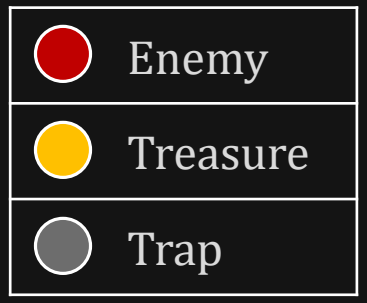
Example gameplay of "Patrol"

5th metric: Guard

- Fitness function:

$$f_{grd} = \frac{1}{\|O\|} \times \sum_{i=1}^{\|O\|} \frac{\frac{\|E\|}{\|O\|} - \left| count(O_i, r) - \frac{\|E\|}{\|O\|} \right|}{\frac{\|E\|}{\|O\|}}, o_j \in O = \{Treasure, Exit\}$$

- Description:
  [會再翻成英文]
  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。

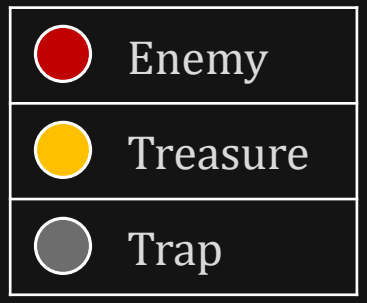


Example gameplay of "Guard"

6th metric: Dominated

- Fitness function:

- Description:
  [會再翻成英文]
  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。

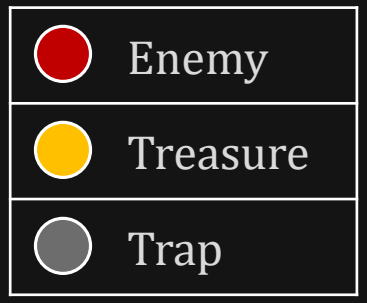


Example gameplay of "Dominated"

7th metric: Support

- Fitness function:

- Description:
  [會再翻成英文]
  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)。

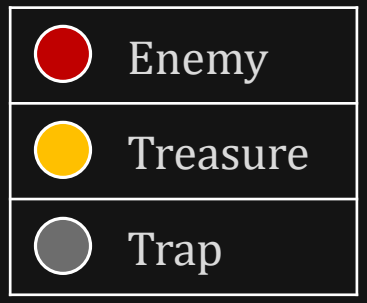


Example gameplay of "Support"

8th metric: Cover

- Fitness function:

- Description:

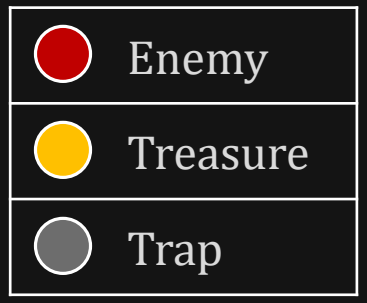


Example gameplay of "Cover"

9th metric: Trap

- Fitness function:

- Description:



Example gameplay of "Trap"

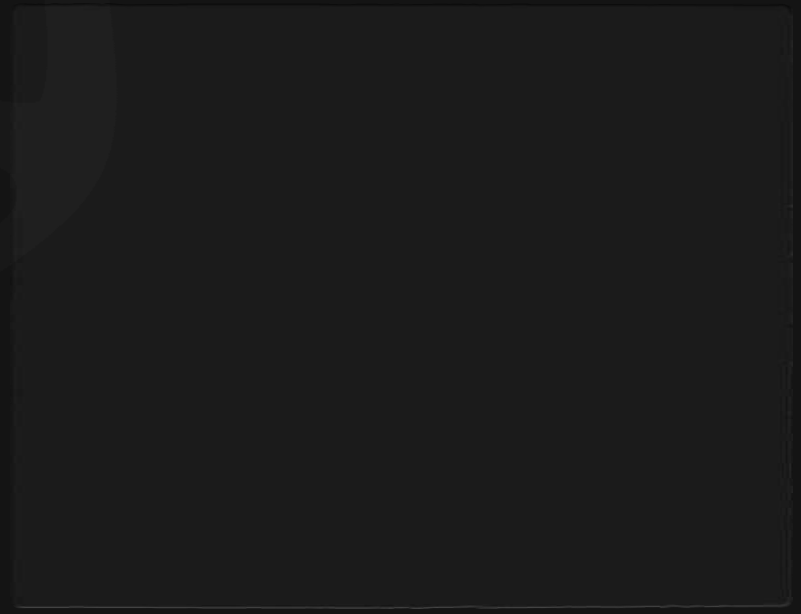
Manual

H Set the weights of fitness functions

- Setting about fitness function weights of each volume
- Same volumes can be duplicated used on different positions

Manual

I Export the completed level



Experimental Results

Experiment design

Data Collection

Run no.	Gen. no.	Chm. no.	label	score	position	type	volume
				. . .			
1	10	3	Block	0	(12.0, 1.0, 0.0)	Enemy	Room A
1	10	3	Intercept	0	(12.0, 1.0, 0.0)	Enemy	Room A
1	10	3	Patrol	0.35	(12.0, 1.0, 0.0)	Enemy	Room A
1	10	3	Guard	0.85	(12.0, 1.0, 0.0)	Enemy	Room A
				. . .			

- Raw data:

Data Filtering of Fitness Comparison

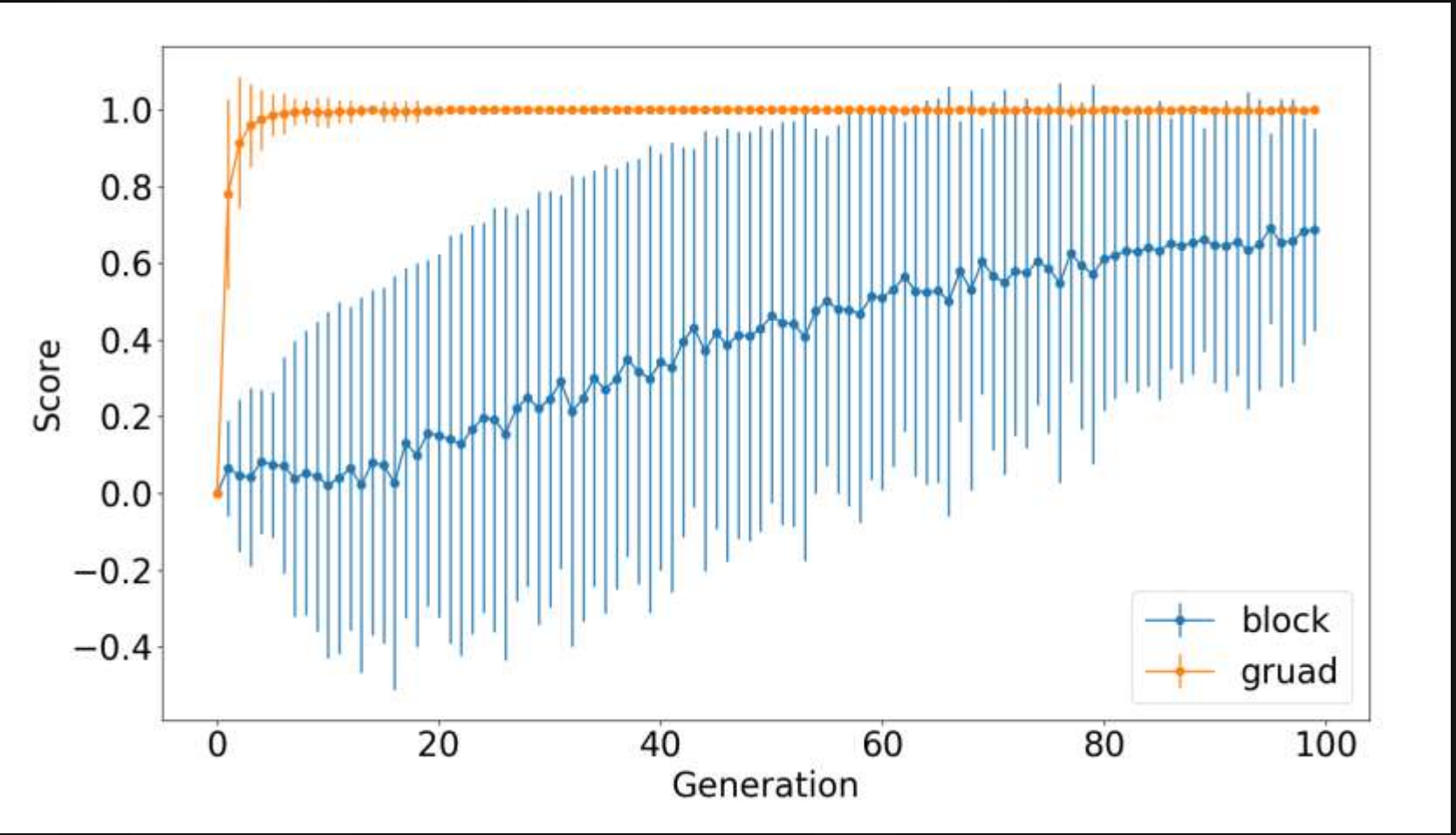
Run no.	Gen. no.	Chm. no.	label	score	position	type	volume
				. . .			
1	10	3	Block	0.35	(12.0, 1.0, 0.0)	Enemy	Room A
1	10	3	Intercept	0.85	(12.0, 1.0, 0.0)	Enemy	Room A
1	10	3	Block	0	(12.0, 2.0, 0.0)	Empty	Room A
1	10	3	Intercept	0	(12.0, 2.0, 0.0)	Empty	Room A
				. . .			

Useless

- Calculate the sum of the fitness score each chromosome.

```
1  // Remove useless columns.
   data.drop(['position', 'type'])
   // Remove duplicated rows.
2  data.drop_duplicates([''])
```

Fitness Comparison



Data Filtering of Density Analysis

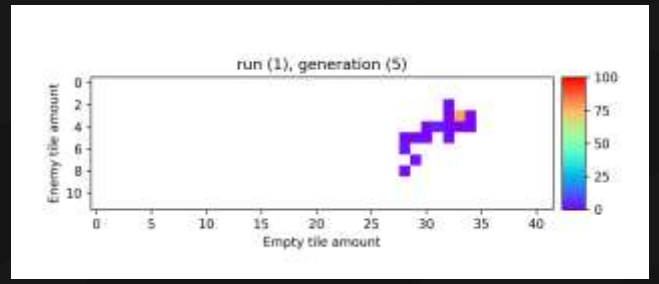
Run no.	Gen. no.	Chm. no.	label	score	position	type	volume
...							
1	10	3	Block	0	(12.0, 1.0, 0.0)	Enemy	Room A
1	10	3	Intercept	0	(12.0, 1.0, 0.0)	Enemy	Room A
1	10	3	Guard	0	(12.0, 1.0, 0.0)	Enemy	Room A
1	10	3	Guard	0.85	(12.0, 1.0, 0.0)	Enemy	Room A
...							

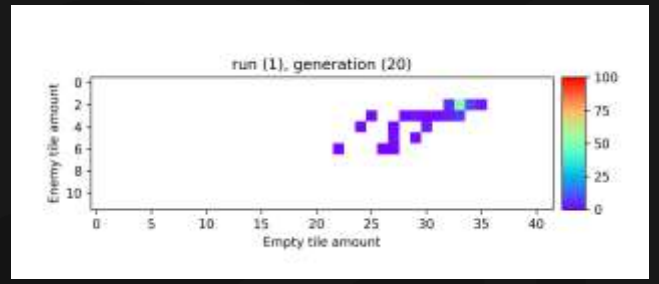
Useless and duplicated

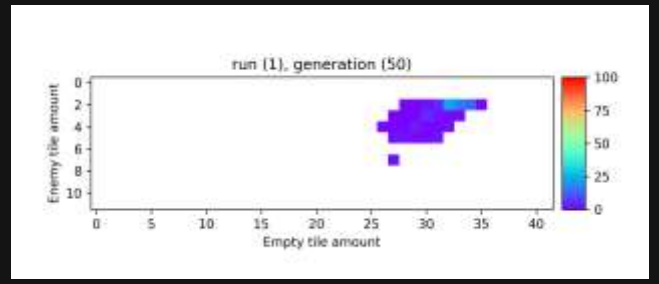
- Store the positions and their gene types each chromosome (individual).

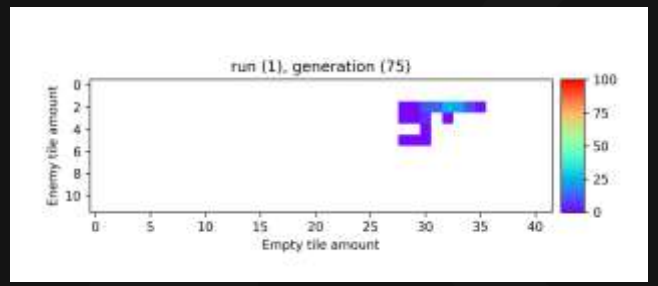
```
// Remove useless columns.
data.drop(['label', 'score'])
// Remove duplicated rows.
data.drop_duplicates(['Run no.',
'Gen no.', 'chm no.', 'position'])
```

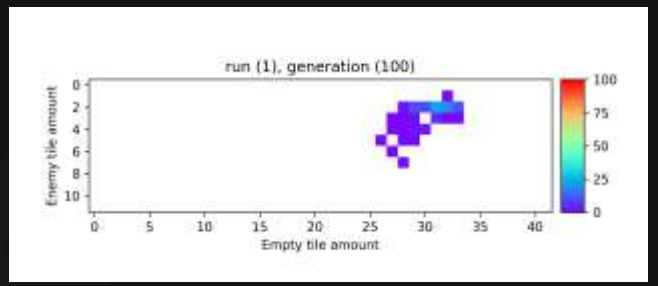
Density Analysis



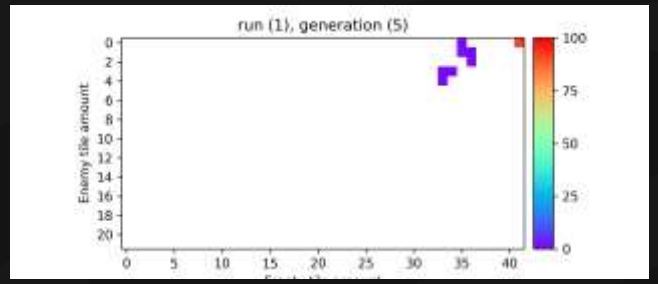


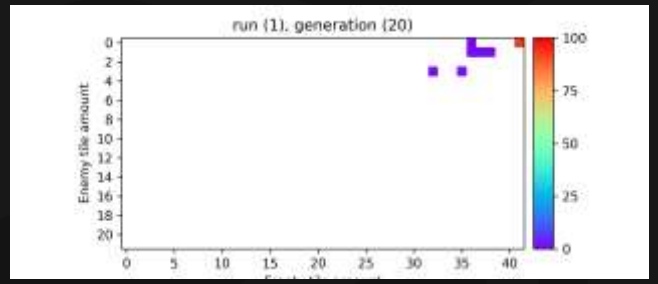


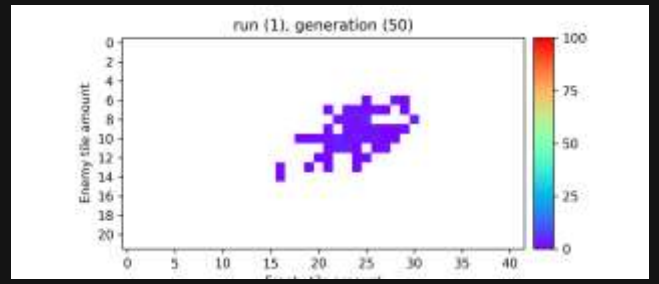


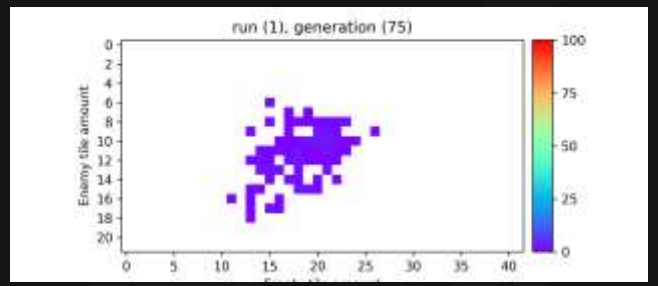


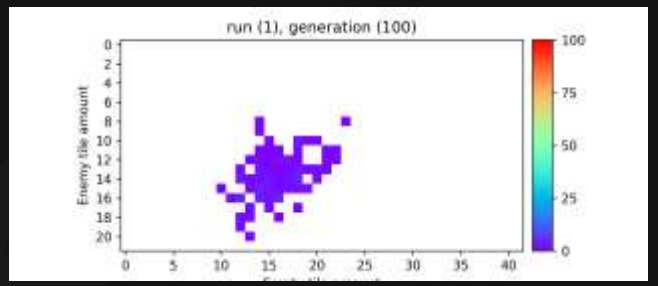
Density Analysis











Conclusions and Contributions

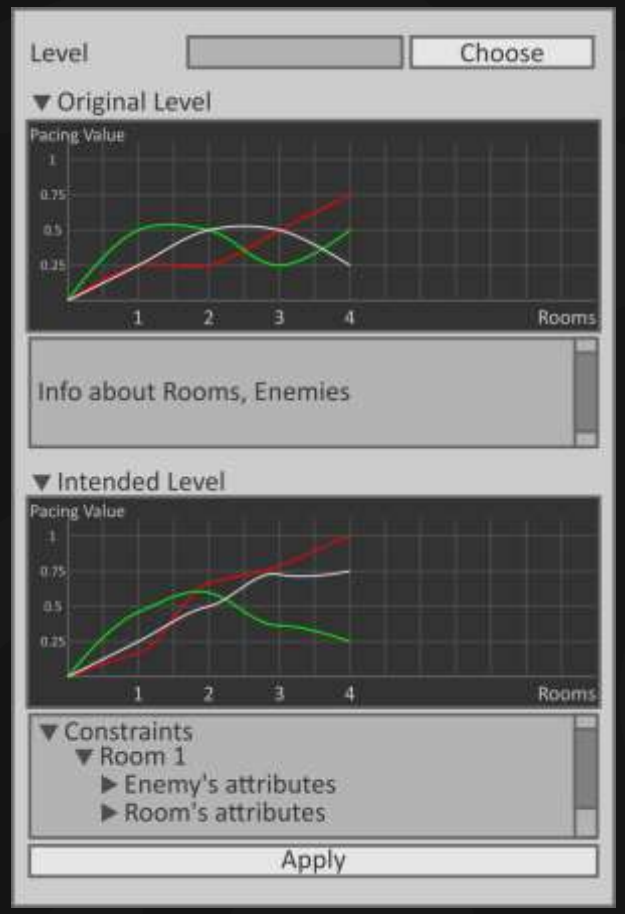
Conclusions

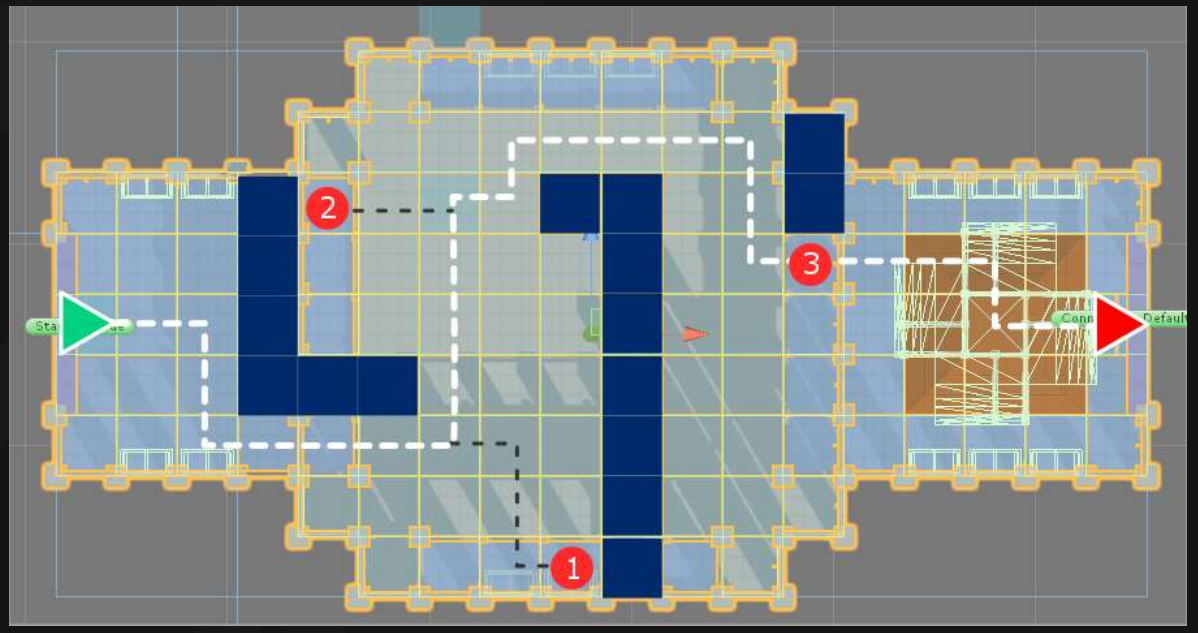
Contributions

- Provide a systematic framework to make game designers develop the game level rapidly.
  - Mission-driven topology.
  - Tool about procedural content generation for mission graph.
- Multi-objective problem in GA

Future Work

Pacing-based Level Generation





References

- Dormans, J. (2010, June). 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.
- Dormans, J. (2011, June). 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.
- Dormans, J. (2012, May). Generating Emergent Physics for Action-Adventure Games. In *PCG@ FDG* (pp. 9-1).
- Karavolos, D., Liapis, A., & Yannakakis, G. N. (2016, September). Evolving missions to create game spaces. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on* (pp. 1-8). IEEE.
- Liapis, A. (2017). Multi-segment Evolution of Dungeon Game Levels.
- Karavolos, D., Bouwer, A., & Bidarra, R. (2015). Mixed-Initiative Design of Game Levels: Integrating Mission and Space into Level Generation. In *FDG*.
- Liapis, A., Yannakakis, G. N., & Togelius, J. (2015). Constrained novelty search: A study on game content generation. *Evolutionary computation*, 23(1), 101-129.
- Liapis, A., Yannakakis, G. N., & Togelius, J. (2013, April). Generating map sketches for strategy games. In *European Conference on the Applications of Evolutionary Computation* (pp. 264-273). Springer Berlin Heidelberg.
- Font, J. M., Izquierdo, R., Manrique, D., & Togelius, J. (2016, March). Constrained Level Generation Through Grammar-Based Evolutionary Algorithms. In *European Conference on the Applications of Evolutionary Computation* (pp. 558-573). Springer International Publishing
- Lanzi, P. L., Loiacono, D., & Stucchi, R. (2014, August). Evolving maps for match balancing in first person shooters. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on* (pp. 1-8). IEEE.
- Zook, A., & Riedl, M. O. (2014, July). Automatic Game Design via Mechanic Generation. In *AAAI* (pp. 530-537).
- Dormans, J. (2012). *Engineering emergence: applied theory for game design.* Creative Commons.
- Neil, K. (2015). *Game Design Tools: Can They Improve Game Design Practice?* (Doctoral dissertation, Conservatoire national des arts et metiers-CNAM).