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

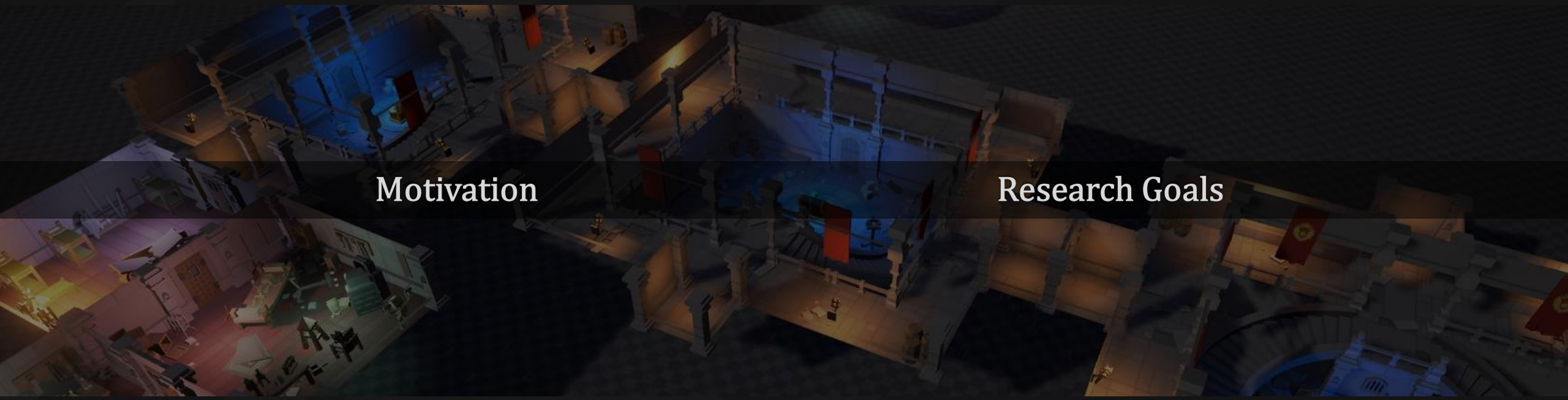
July 27, 2017

Prof. Wen-Kai Tai	Prof. Hsing-Kuo Pao	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
  - Build Volumes
  - Generate Space
  - Volume Evolution
- Experimental Results
- Conclusions and Contributions
- Future work

Introduction



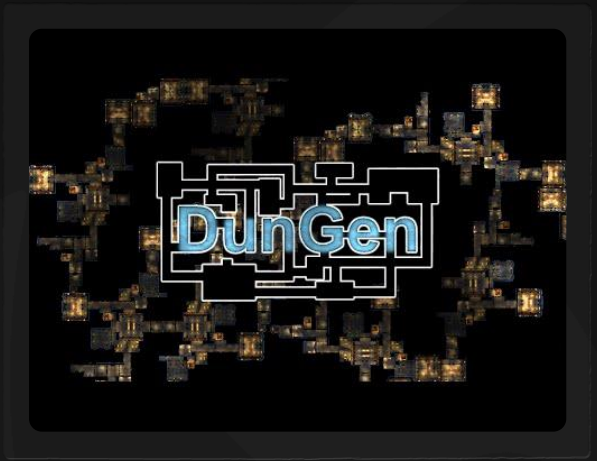
Motivation

Research Goals

Motivation

- 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。

Dungeon Architect

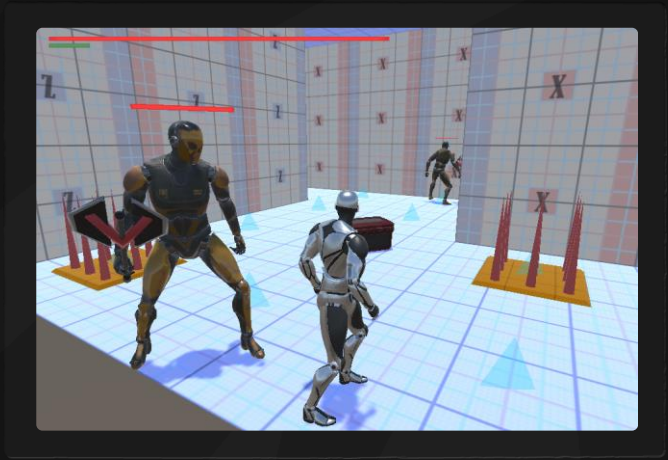


DunGen

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

Research Goals

- 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 spatial topology	PCG	Randomly PCG, Manually	Randomly PCG
Mission-based topology	Dynamic-nonlinear, meaningful structure	None	Stable-linear structure
Game Patterns	Genetic Algorithm	None	None

Research Goals (Cont'd)

- Referenced Level Generation Methods
  - Joris Dormans – Mission/Space framework
  - Antonios Liapis – Sentient Sketchbook

Method / Framework	**Mission progress**	**Evaluate content**
Mission / Space framework	Accurately control	None
Sentient Sketchbook	None	Provide via GA

Related Works

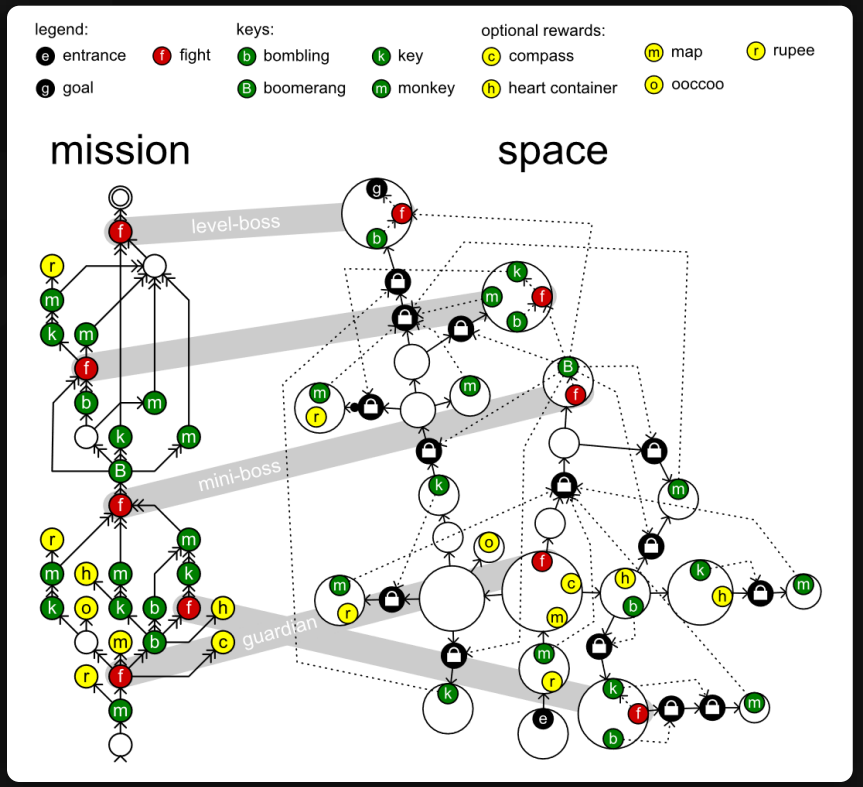
Mission / Space framework

Map Sketches & Evolution of Segments

Mission / Space framework

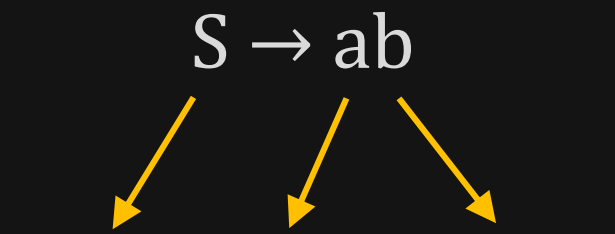
Mission/Space framework, focuses on **level design** and the **mechanics** that control player progression through a game.

- Transformational Grammars
- Mission Grammars
- 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

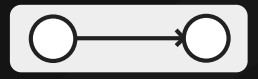
Mission Grammars

-  **Inhibitions**

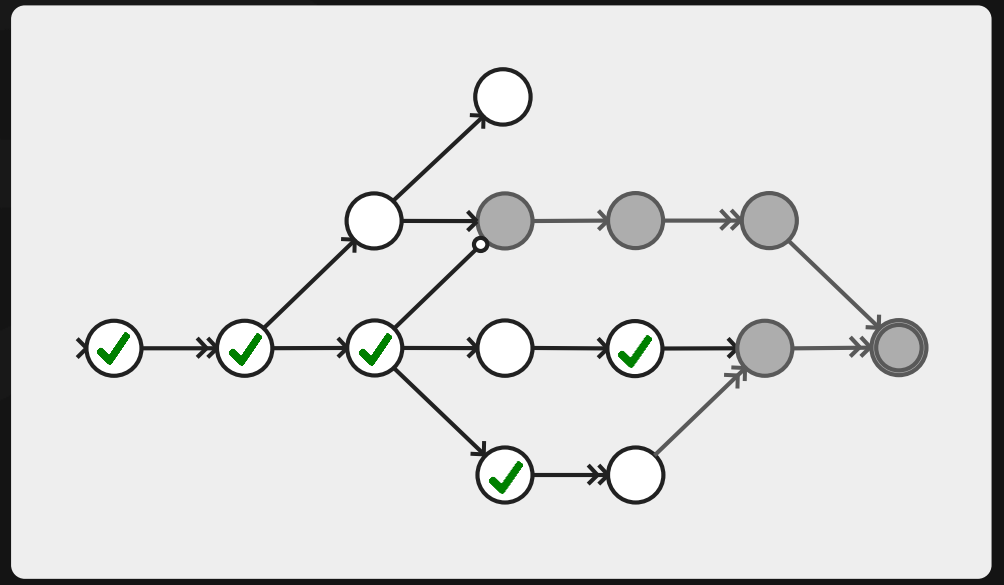
  Available when at least one of the weak prerequisites is completed.

-  **Strong requirement**

  Available when all strong prerequisites are completed.

-  **Weak requirement**

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

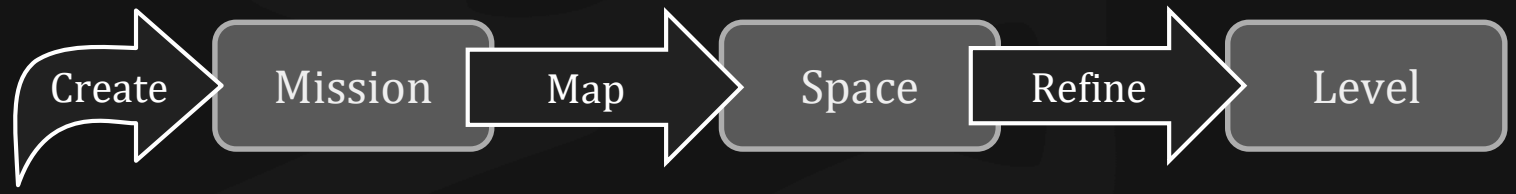


Example of mission graph

- 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.

Mission Graph Convert into Space Graph

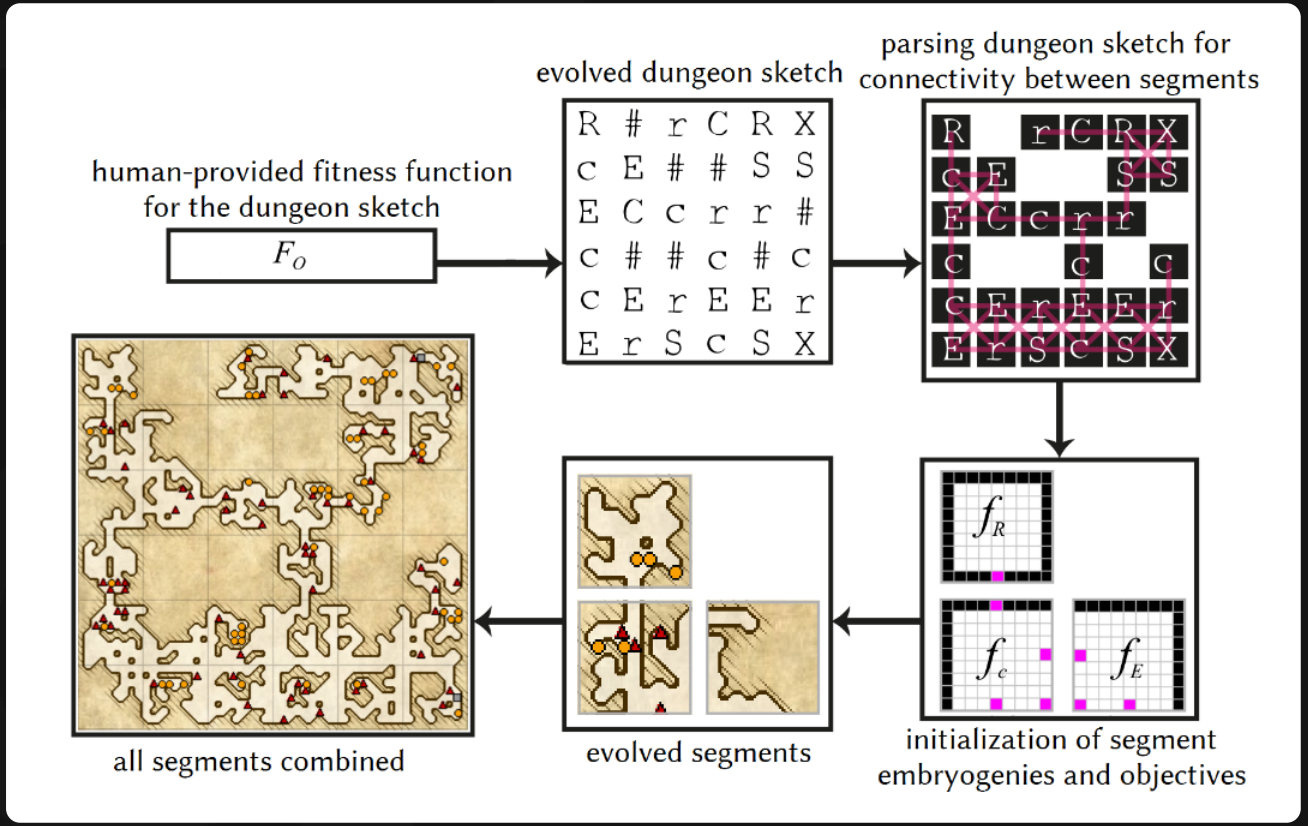
- The steps in the generation progress investigated in this paper in detail:



- An alternative method, does not go into detail:

Map Sketches & Evolution of Segments

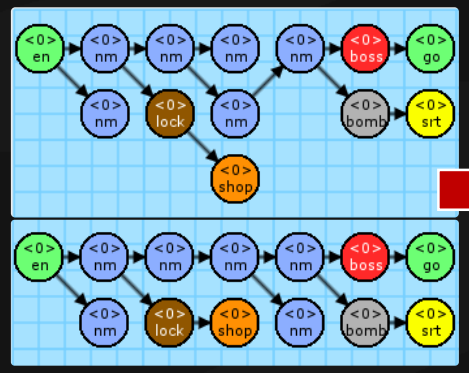
- Map Sketches
- Map Sketch Evolution
- Dungeon Segments
- Dungeon Segment Evolution



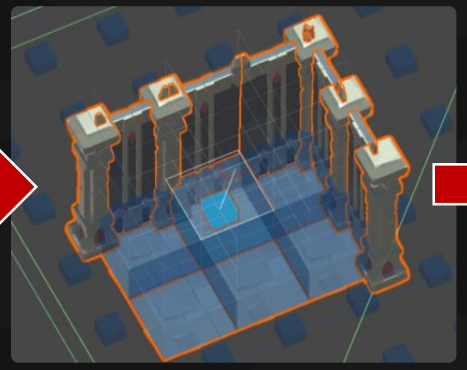
Proposed Methodologies

1. System Framework
2. Mission Grammars
3. Build Volumes
4. Generate Space
5. Volume Evolution

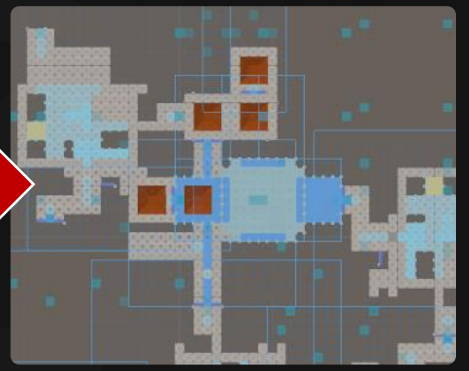
System Framework



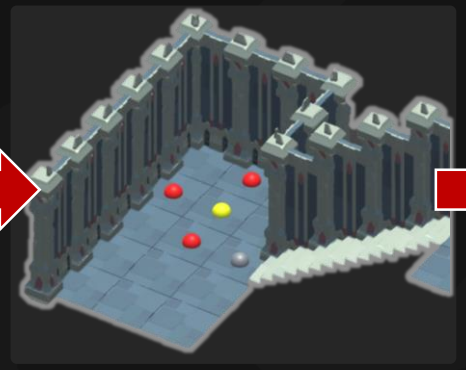
Mission grammars



Build volumes



Generate Space



Volume Evolution

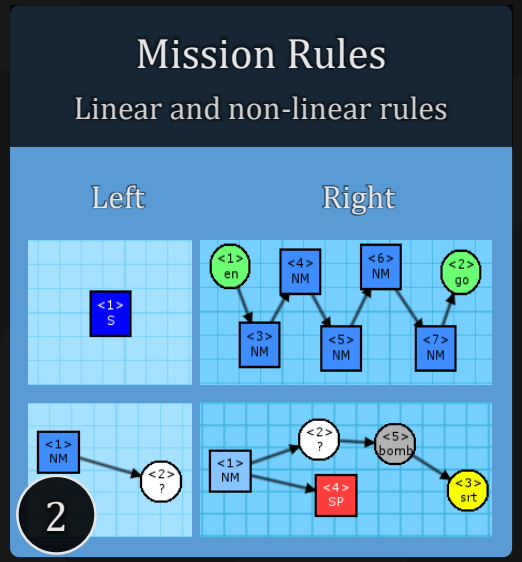
Completed level

System Framework

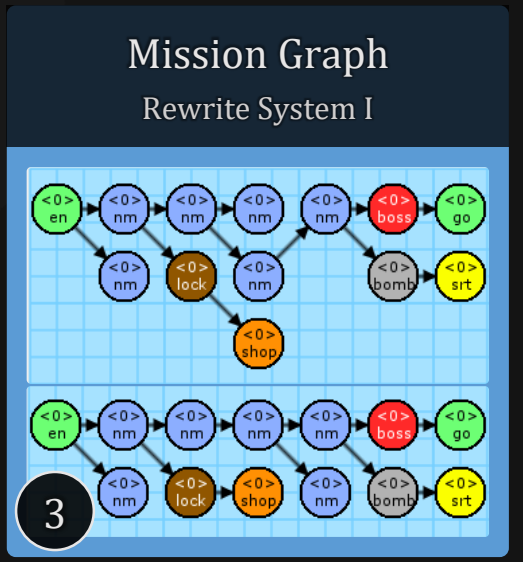
1. Mission Grammars

Mission Alphabet
Extract game patterns

Mission Rules
Linear and non-linear rules



Mission Graph
Rewrite System I

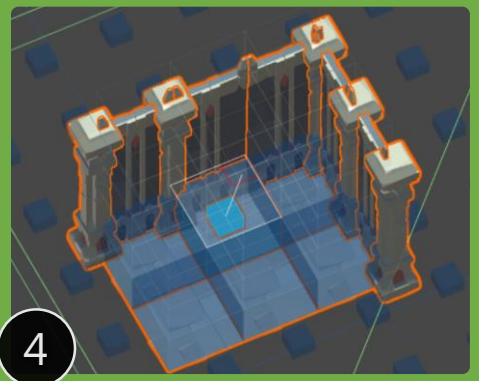


System Framework

2. Build Volumes

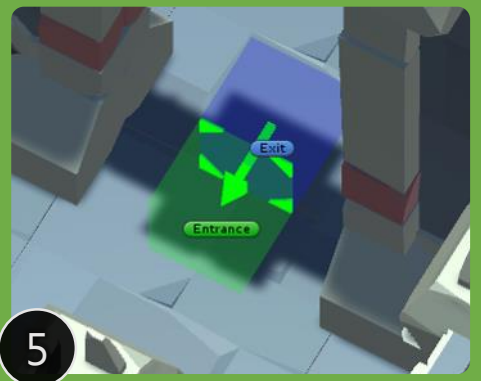
Volume

Voxel-based and decorations



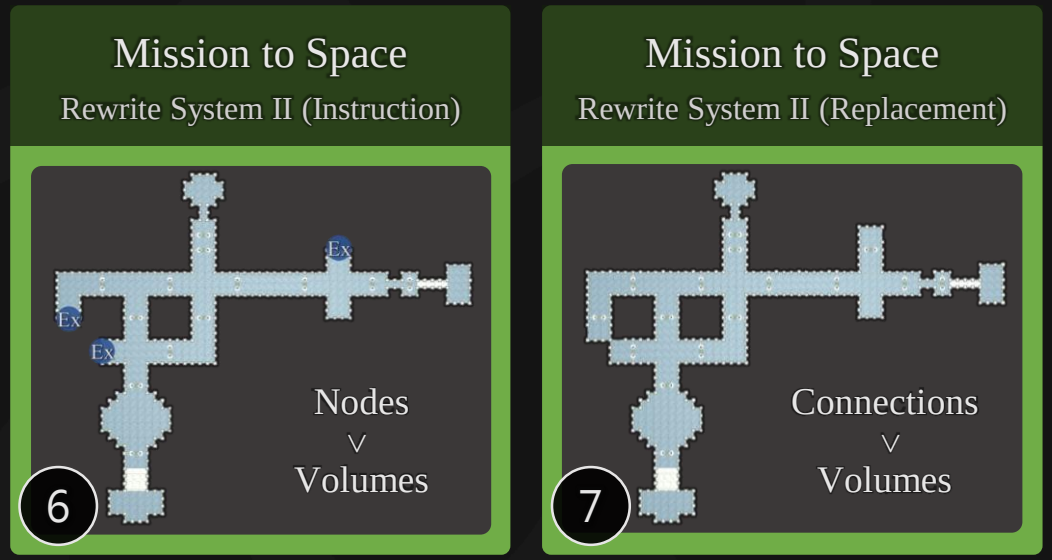
Connection

Entrance and Exit markers



System Framework

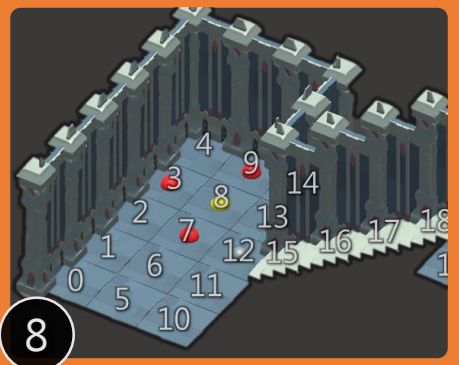
3. Generate Space



System Framework

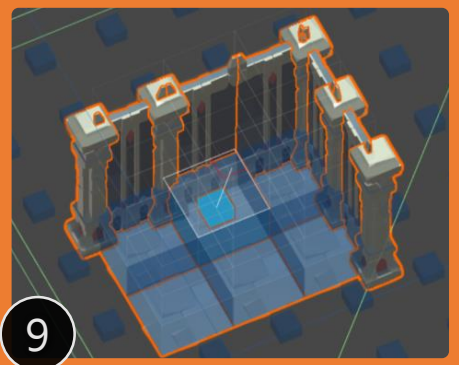
4. Volume Evolution (Generate the Gameplay Patterns)

Genes
Passable tiles



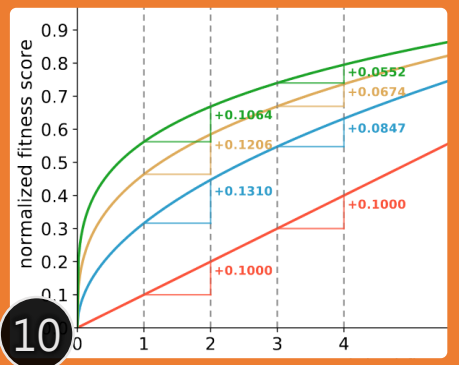
8

Fitness functions
Design the metrics



9

Normalization
Control the amplitude



10

Evolution
Adjust weights of fitness functions

11

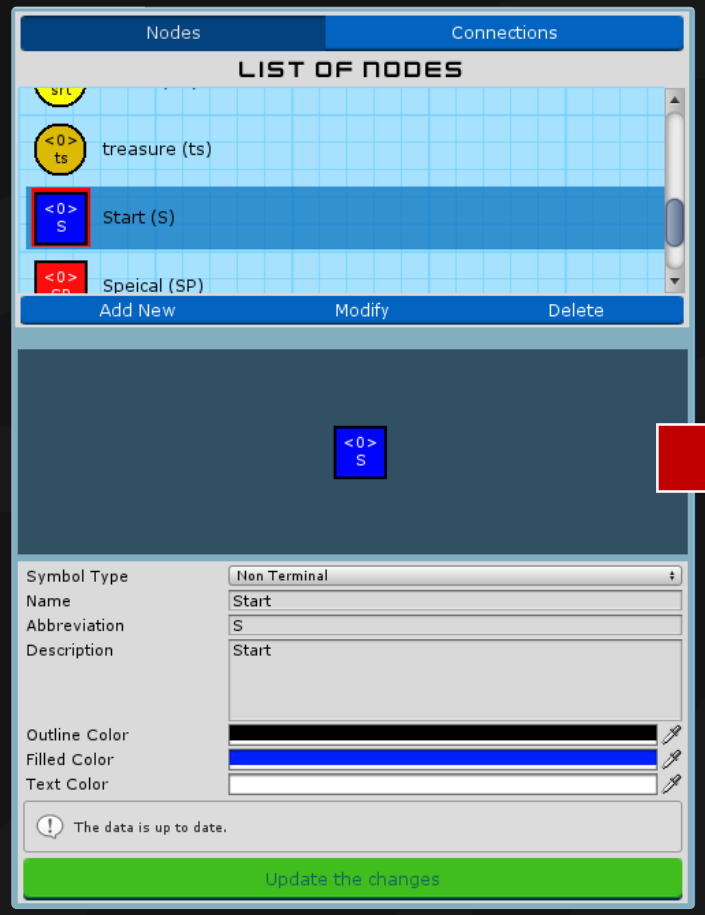
System DEMO

Video here

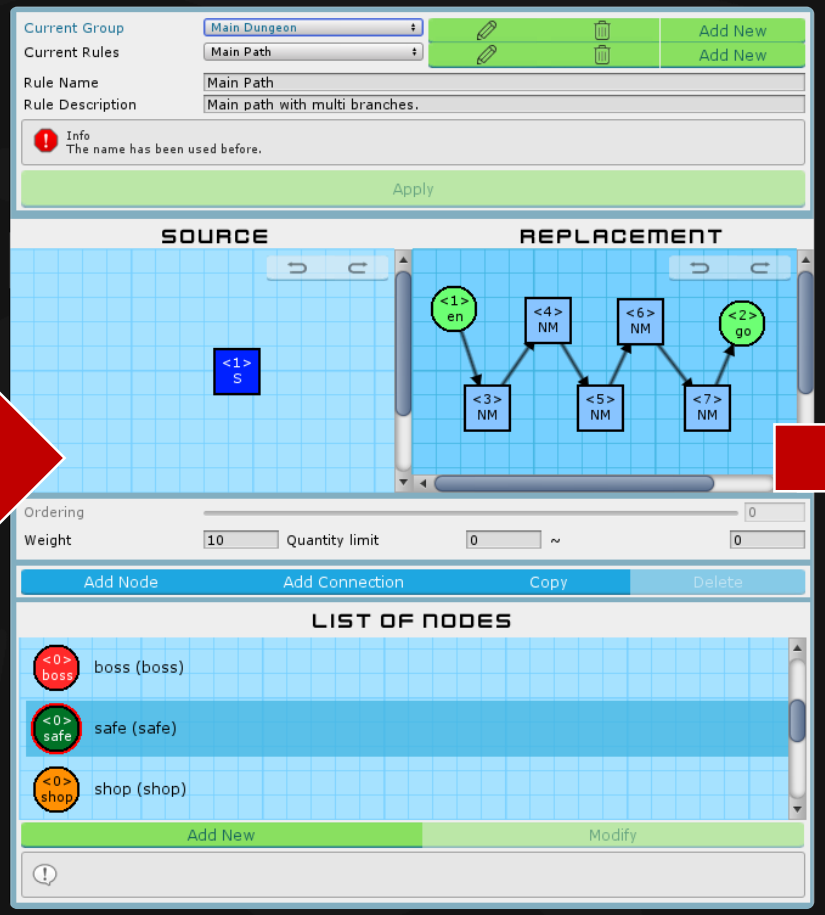
Section 1. Mission Grammars

Export the mission graph

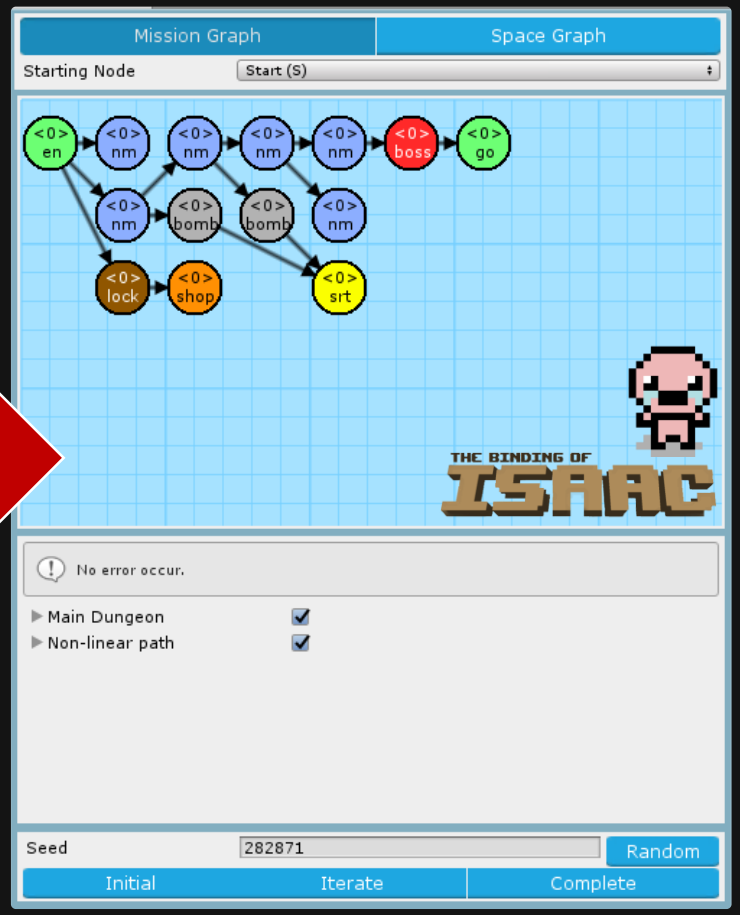
Mission Grammars



Mission Alphabet

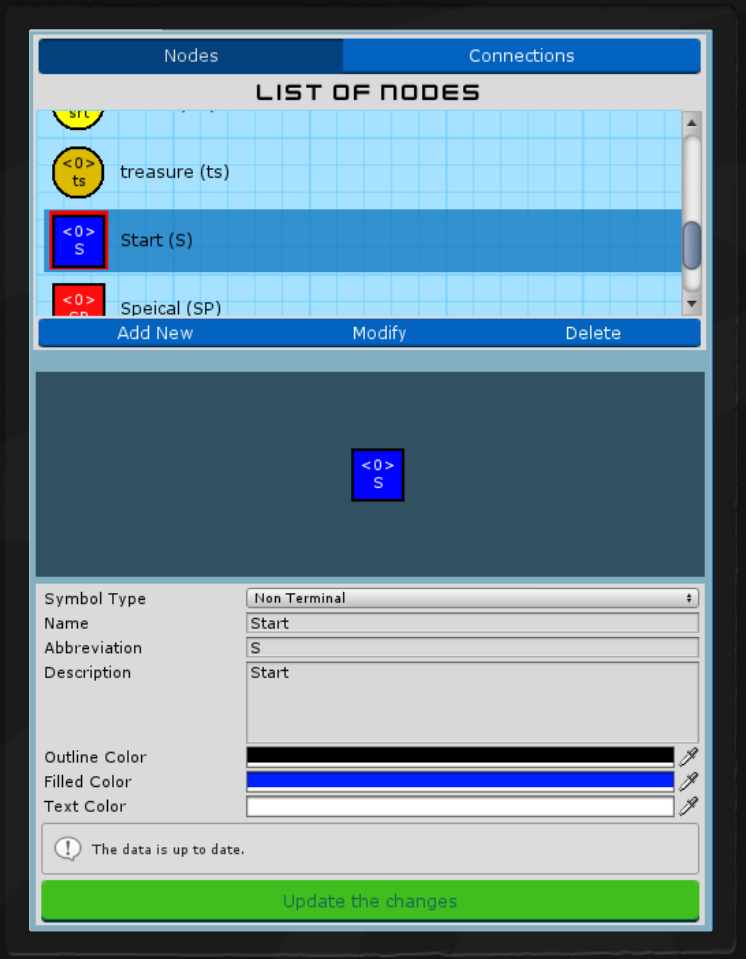


Mission Rules



Mission Graph

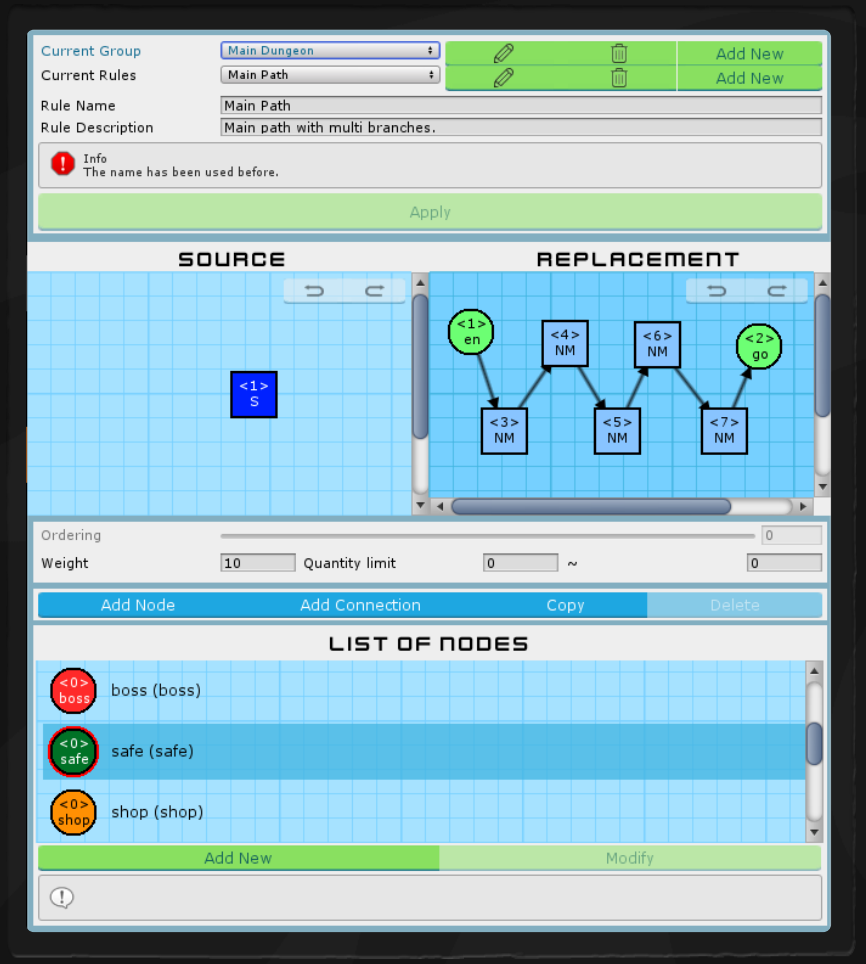
Introduction of Mission Alphabet



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: *Entrance*, *Goal*
  - Extended: *Any*

Introduction of Mission Rules

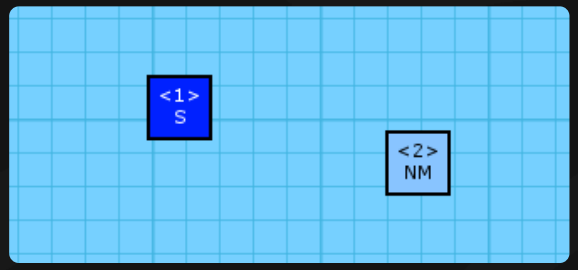


Mission Rules Window

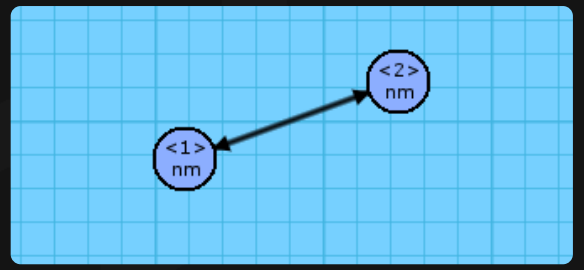
- Hierarchy structure
  - Groups > Rules > Graph Grammar of Left-hand side & Right-hand side Rules
  - Friendly interface to create, delete and edit.
- Rule canvas
  - Drag & drop to modify symbols
  - Custom size of canvas
  - Selected symbol highlighting
  - Connections stick on nodes automatically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

Avoid the illegal mission rules

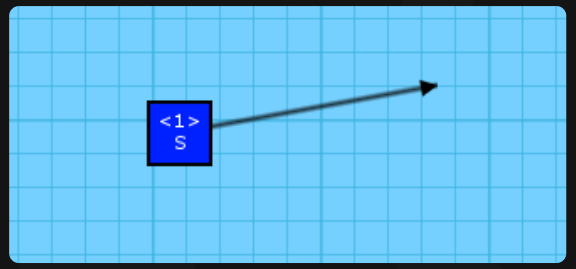
- Illegal rules will lead to the rewrite system generating fail or encounter the loop.



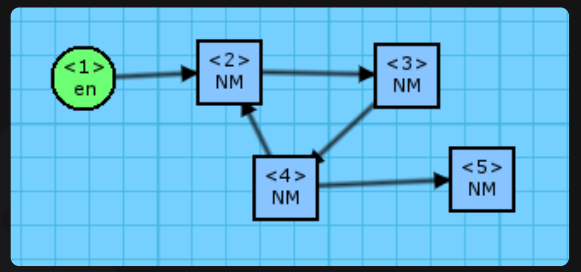
Isolated Node



Multiple Relations



Isolated Connection



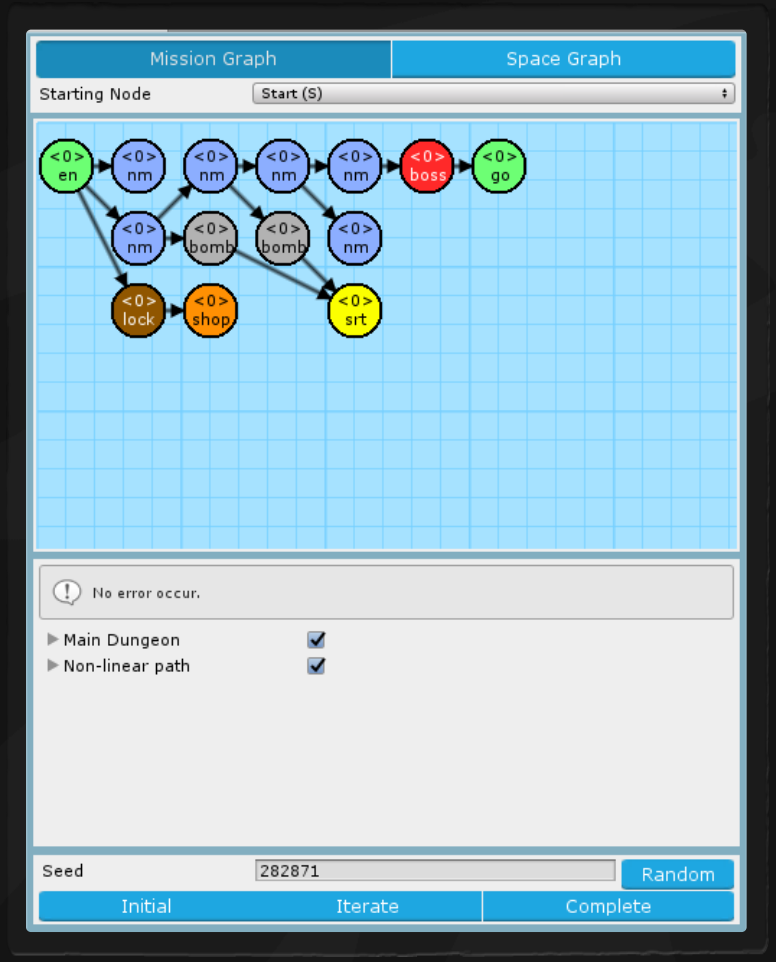
Cyclic Link

And else ...

Set of 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]

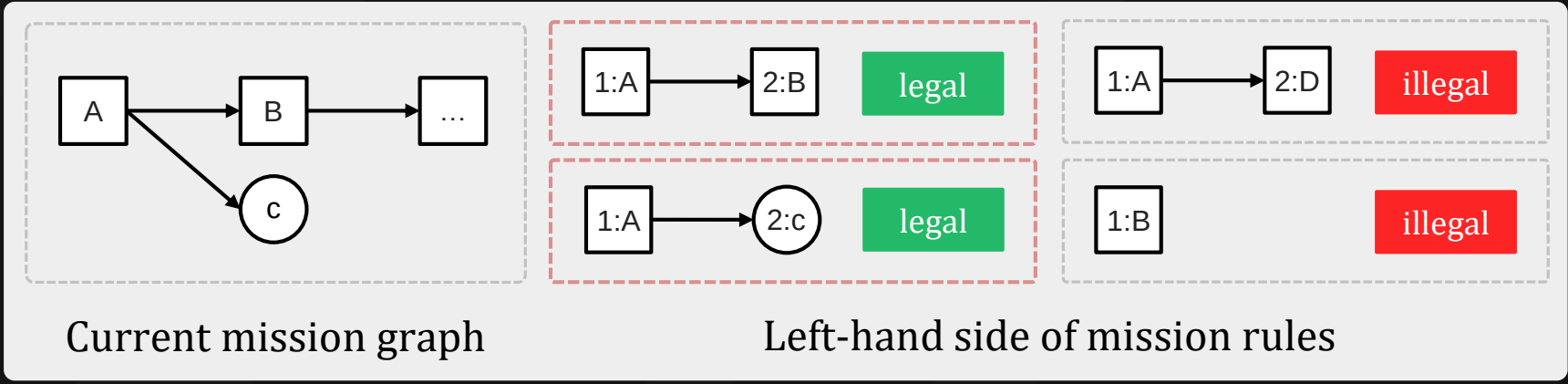
---

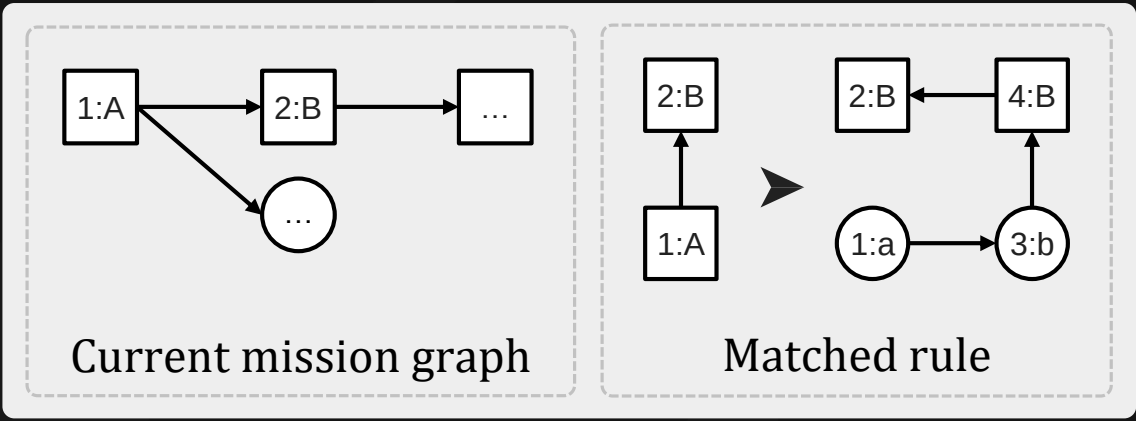
[1] VF Graph Isomorphism Algorithm: https://www.codeproject.com/Articles/23144/A-C-Implementation-of-the-VF-Graph-Isomorphism-Alg

Rewrite System I

- Find the matched rule from all rules
  - Randomly pick one if found many rules.
  - **Current mission graph** and **left-hand side of rule** are subgraph isomorphic.
- Update the mission graph from the matched rule
  - Iterating until all nodes are replaced to terminal nodes.

“FindMatchs” works

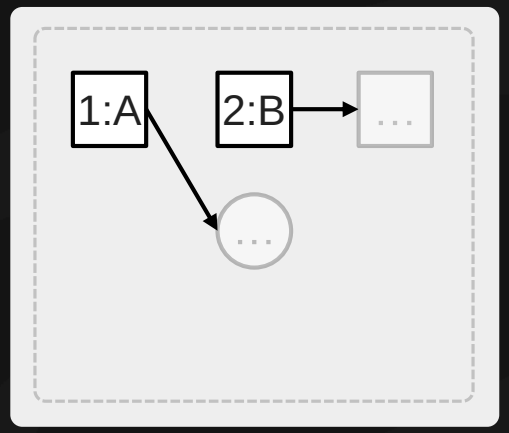




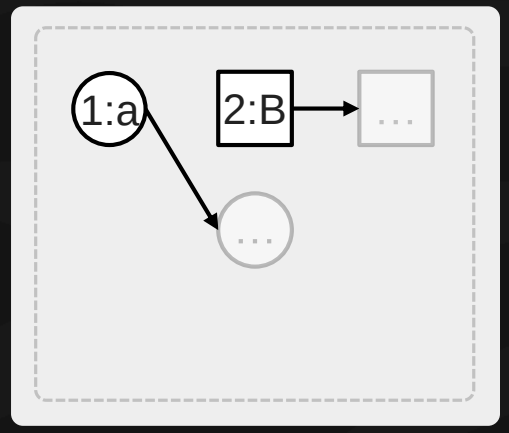
Pseudocode of FindMatchs

```
1  function FindMatchs(root, rules)
2      matchRules is an empty array of rule
3      graph = TransformToGraph(root)
4      for all rule ∈ rules do: // Based on mission rule table
5          if graph is isomorphic with left of rule then:
6              matchedRules add rule
7          end if
8       end for
9       selectedRule = RouuletteWheel(matchedRules) // Selection is based on random table.
10      SetIndexes(root, selectedRule)
11      return selectedRule
12 end function
```

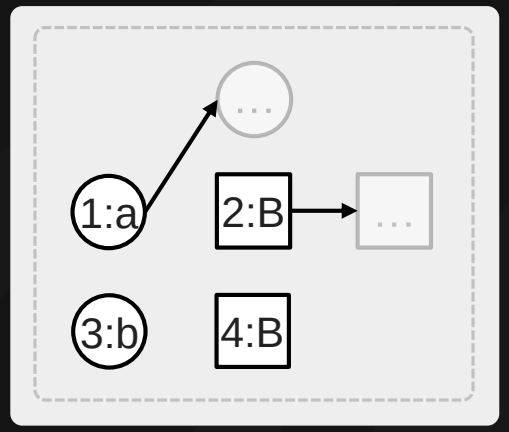
“Rewrite System I” works



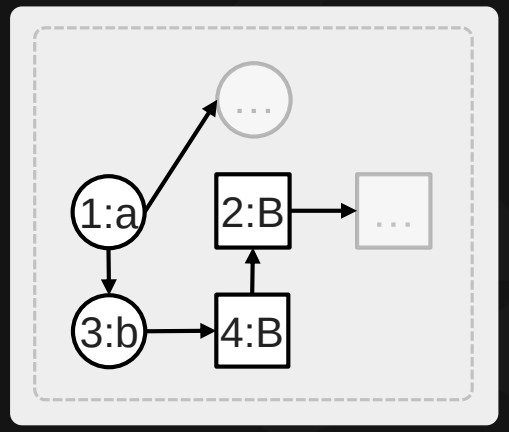
1. RemoveConnections



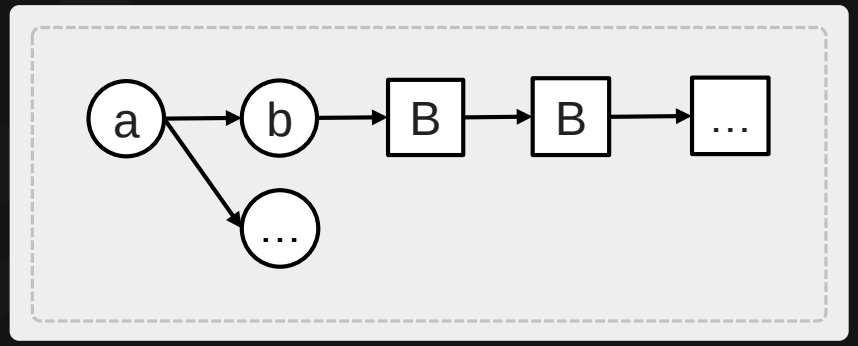
2. ReplaceNodes



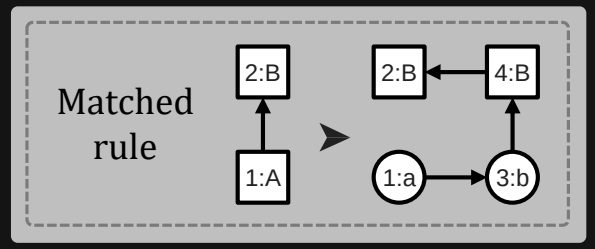
3. AppendNodes



4. ReAddConnections



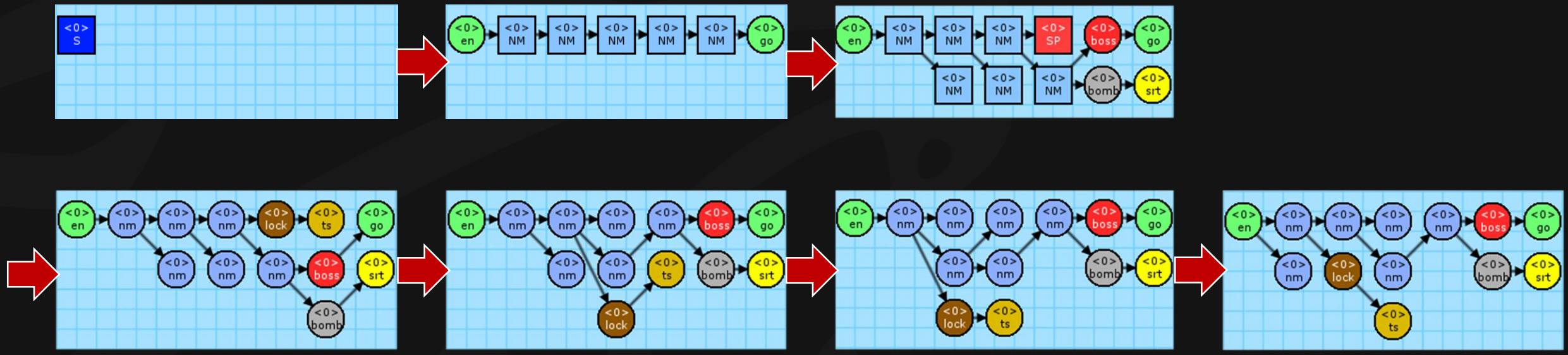
5. RevokeIndexes (Final)



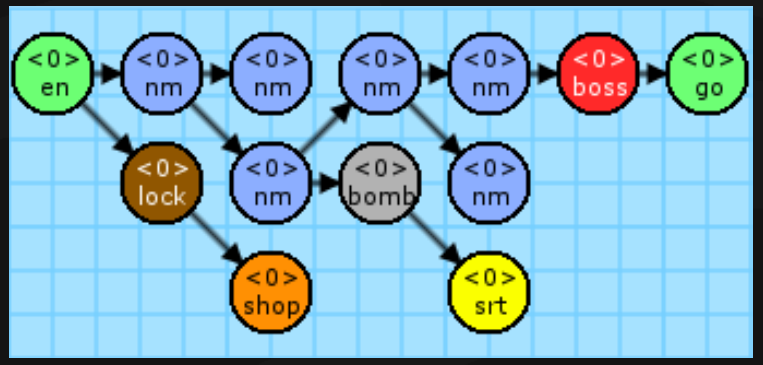
Pseudocode of Rewrite System I

```
1  Initialize random number generator via seed // Seed will affect the selection in FindMatchs
2  function RewriteSystem1(root, rules)
3      matchedRule = FindMatchs(root, rules)
4      if matchedRule is found from root and root doesn't be explored then:
5          RemoveConnections(root, matchedRule) // Remove the connections of matched parts.
6          ReplaceNodes(root, matchedRule)      // Replace the nodes based on right-hand side
7          AppendNodes(root, matchedRule)       // Append the nodes that additional index in right-hand
8          ReAddConnections(root, matchedRule)  // Re-add the connections based on right-hand side
9          RevokeIndexes(root, matchedRule)     // Revoke the index information of nodes in graph
10     end if
11     for all child ∈ children of root do: // DFS
12         RewriteSystem1(child, rules)
13     end for
14     return root
15 end function
```

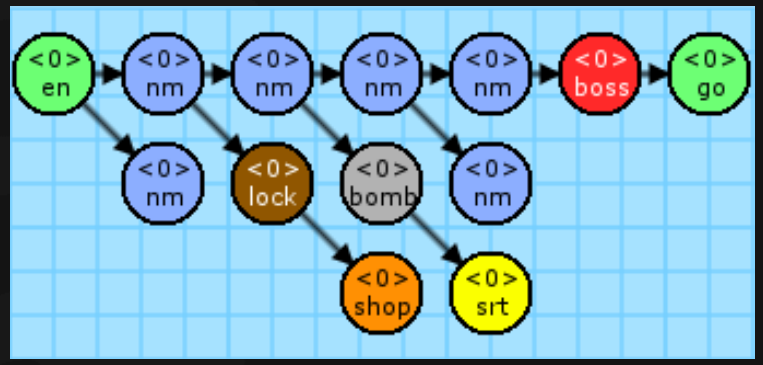
Perform iteration



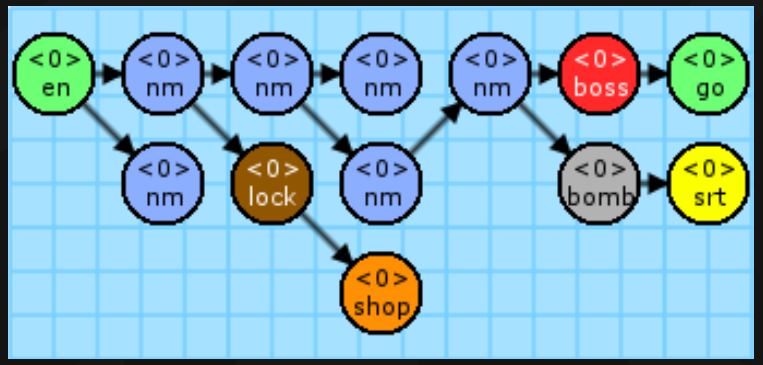
Example of Mission Graph



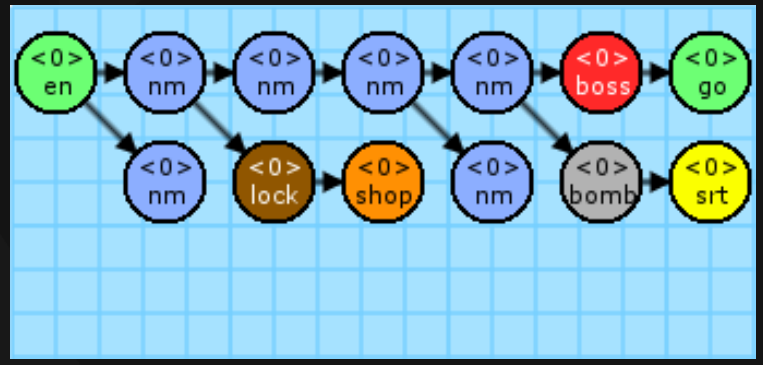
(a) Seed = 653543



(b) Seed = 246617



(c) Seed = 185606



(d) Seed = 183434

Manual

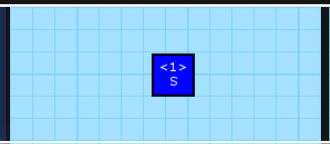
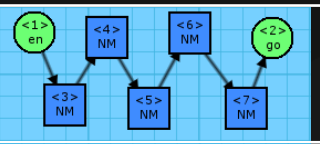
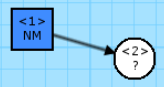
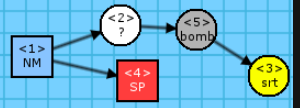
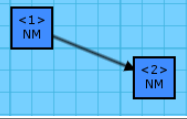
A Create the Mission Alphabet

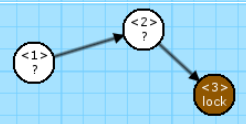
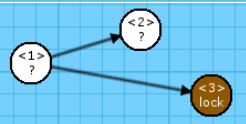
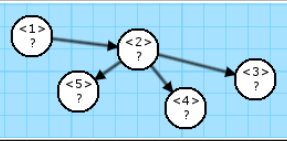
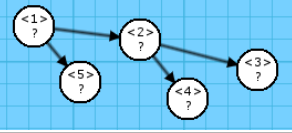
Node	Start (S)	entrance (en*)	Normal (NM)	normal (nm)	boss (boss)	goal (go*)	Special (SP)	shop (shop)	bomb (bomb)	lock (lock)	secret (srt)	treasure (ts)	any (?*)
Display													
Require-ment					Battle and end		Multi - choice	Get the key from secret, then enter the secret or treasure room behind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

** Means the nodes are system default nodes.*

Manual

B Design the Mission Rules

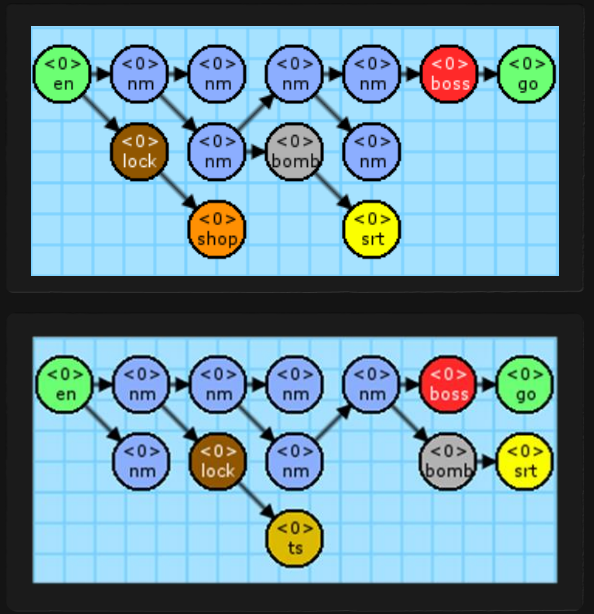
	Left-hand	Right-hand	
Main Path			Weight: 10 Limit: [0, ∞]
Boss Room			Weight: 150 Limit: [0, 1]
Set Secret			Weight: 30 Limit: [1, 1]
Expora-tion			Weight: 10 Limit: [0, ∞]
More Branch			Weight: 20 Limit: [0, 2]

	Left-hand	Right-hand	
Shop			Weight: 10 Limit: [0, 1]
Treasure			Weight: 10 Limit: [0, 1]
Forward Lock			Weight: 10 Limit: [0, 2]
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.



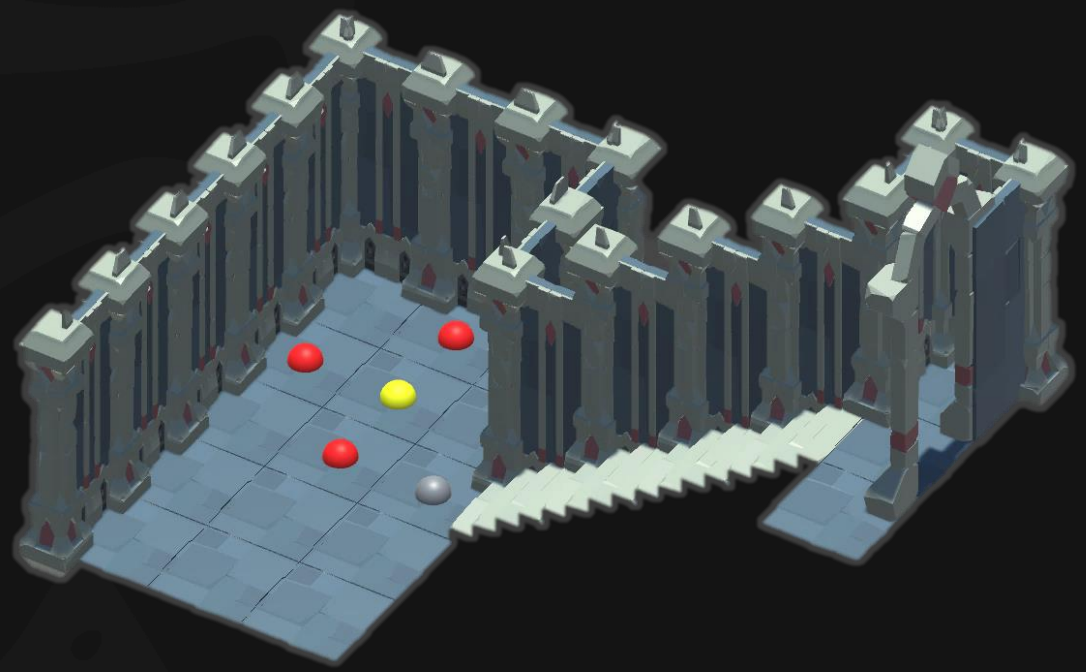
[1] additional iterations: The completed generation means all nodes in the graph are replaced to terminal nodes. Additionally executes more iterations to parallel rewrite the current graph.

Section 2. Build Volumes

Build the volumes of level

Build Volumes

- Editor
  - Voxel-based units
  - Markers I - Decorations
  - Markers II - Connections



Voxel-based units in Editor

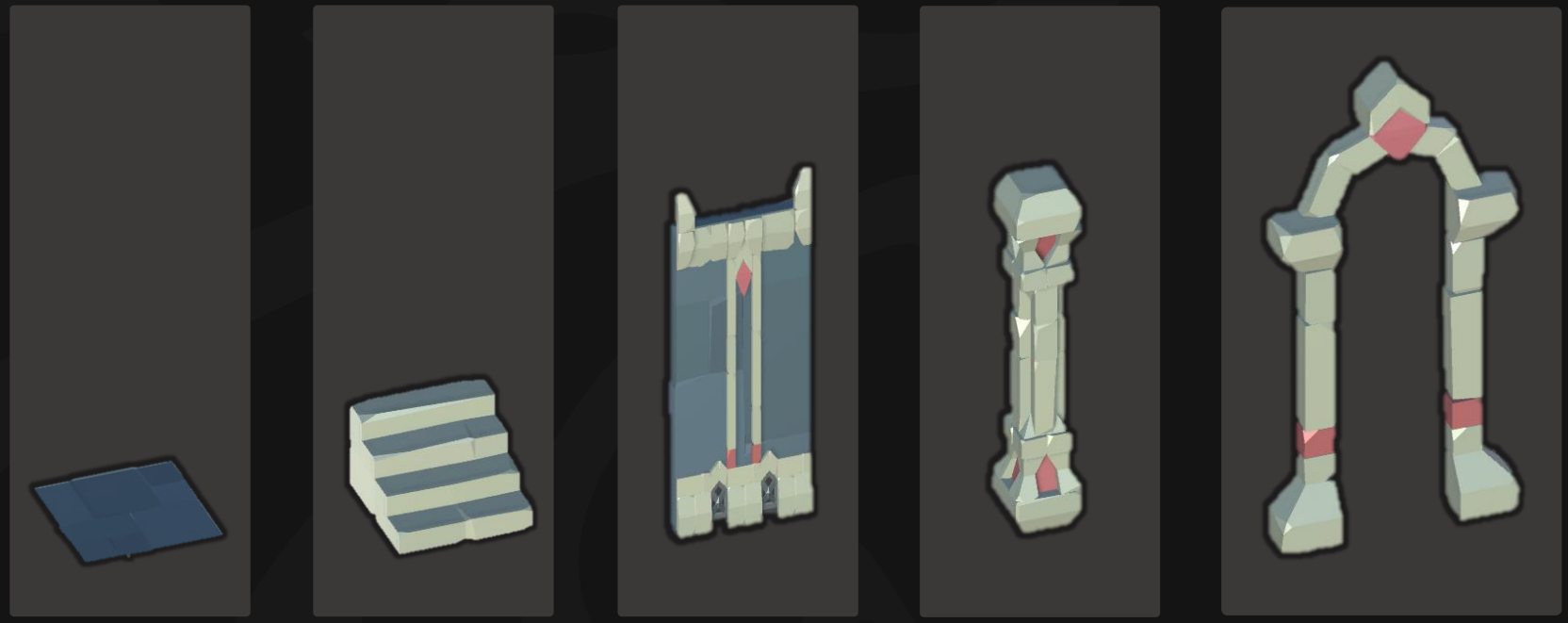
- Hierarchy data structure
  - **Level**
    A set of volumes. Expresses a complete game level.
  - **Volume**
    A set of chunks. Mostly expresses a room.
  - **Chunk**
    Consists of 9x9x9 voxel. Based on the size of volume, uses different numbers of chunk.
  - **Voxel**
    The minimum unit of level,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

Floors

Walls

Pillars

Decorations in Editor

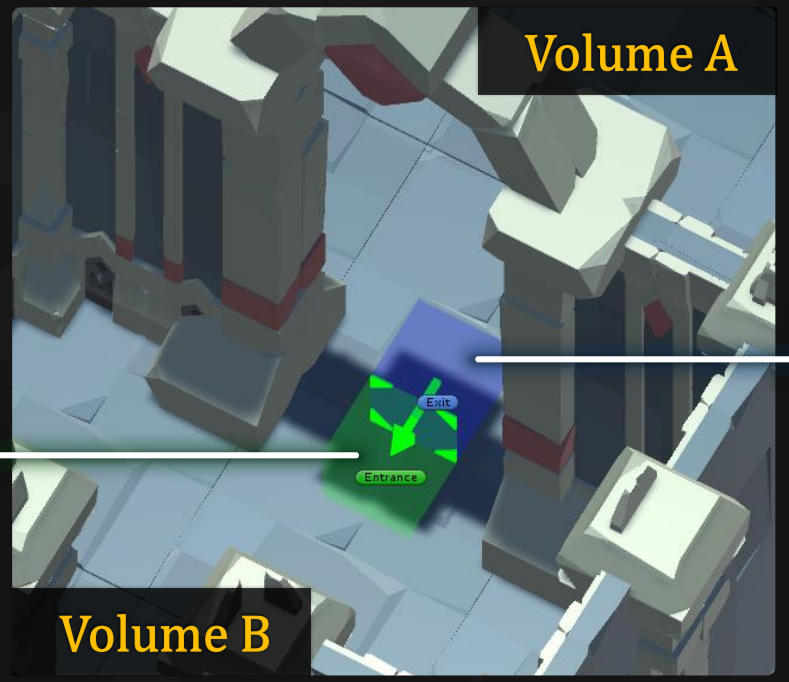


Floor *Stair* *Wall* *Pillar* *Door*

Connections in Editor

Entrance

- Zero or one
- Inner direction



Exit

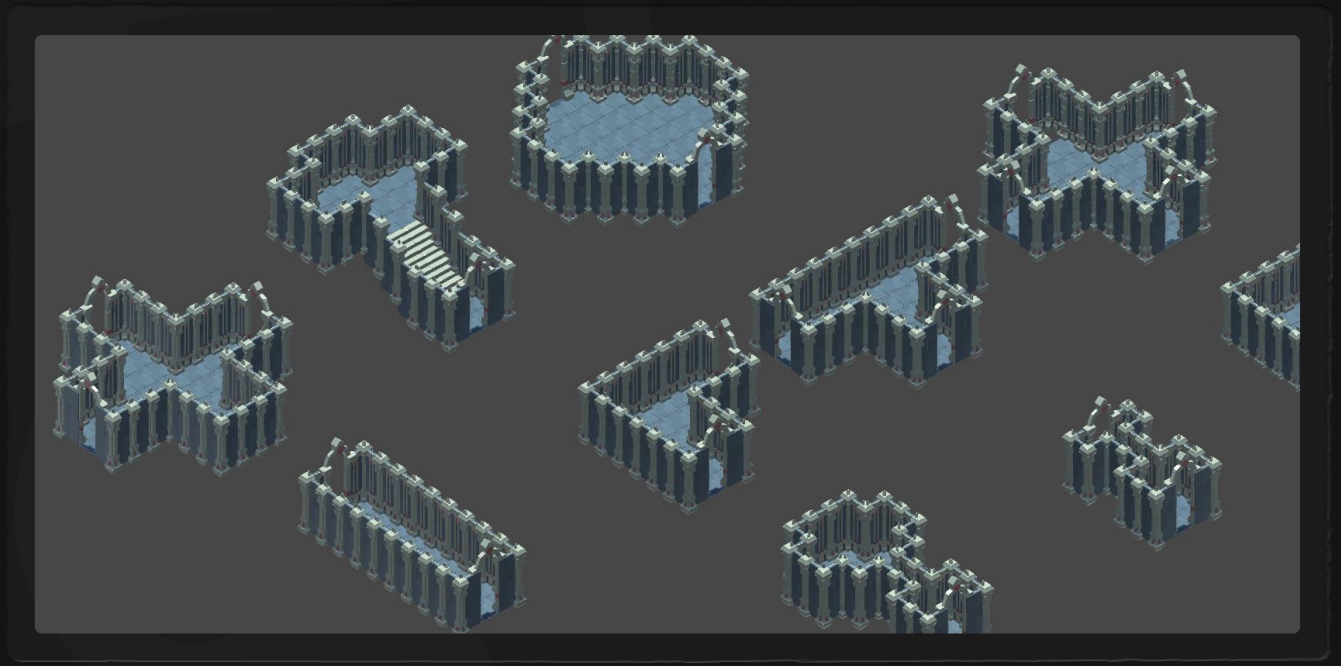
- Any amount
- Outer direction

- 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.

Manual

D Design the Volumes

- Must prepare the volumes to map the mission alphabet.
- Must exist one entrance or exit at least each volume.

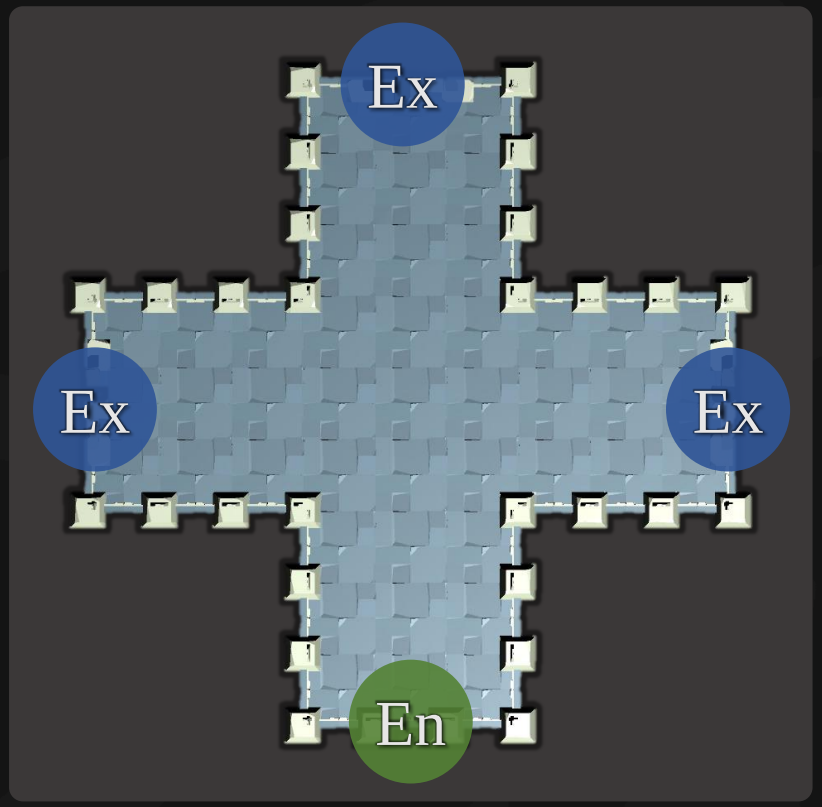


Manual

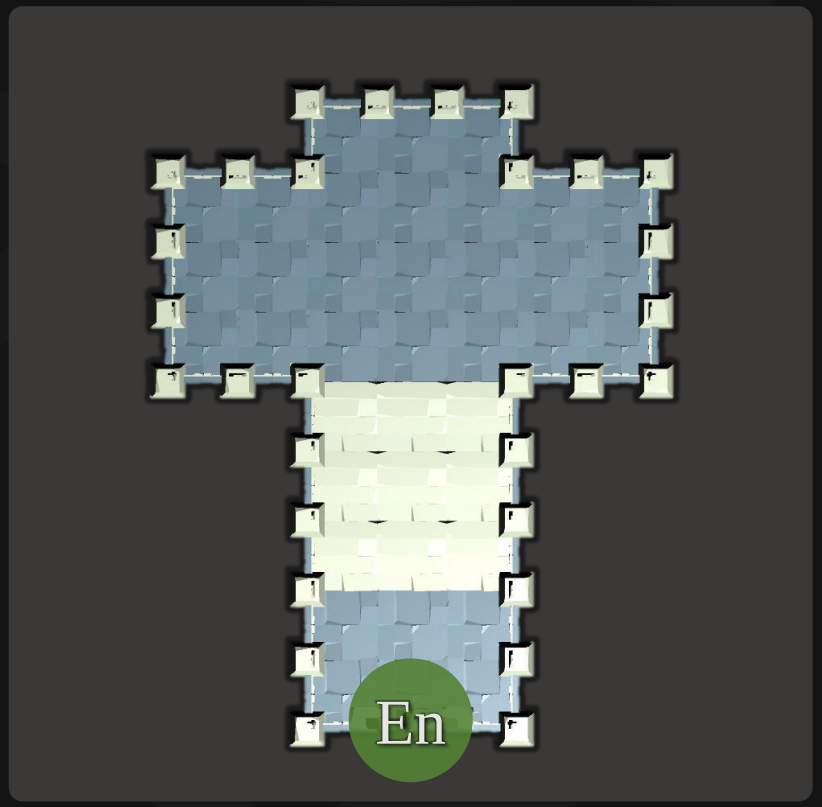
Append the Entrance & Exit

- Append the connection marker with direction
- Binding will based on their y-axis layer
- Stochastic one exit will be an entrance if don't exist entrance marker in the volume

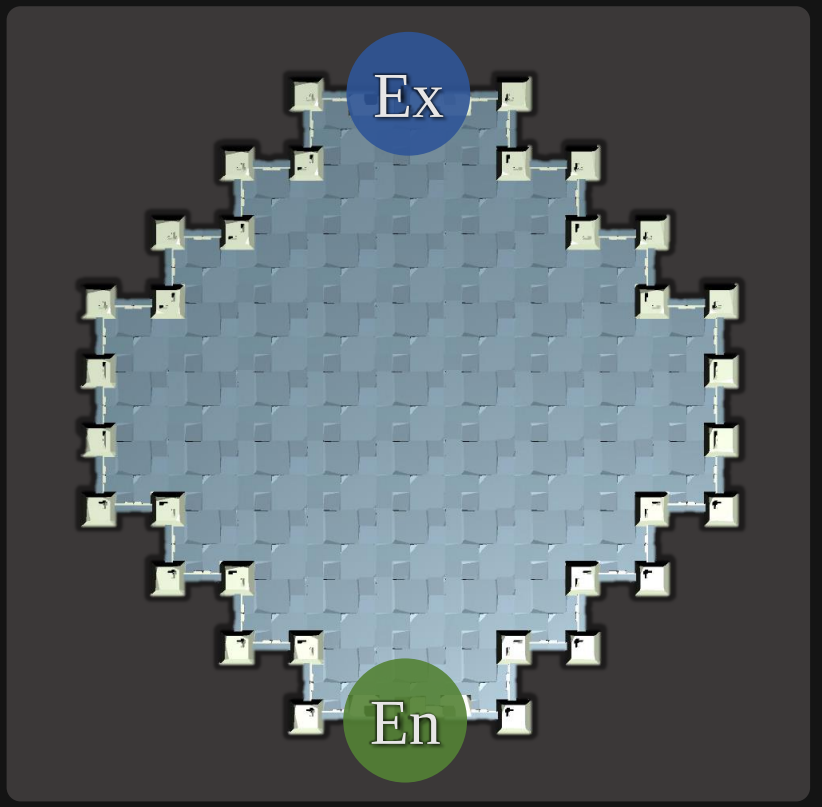
Volumes List



entrance

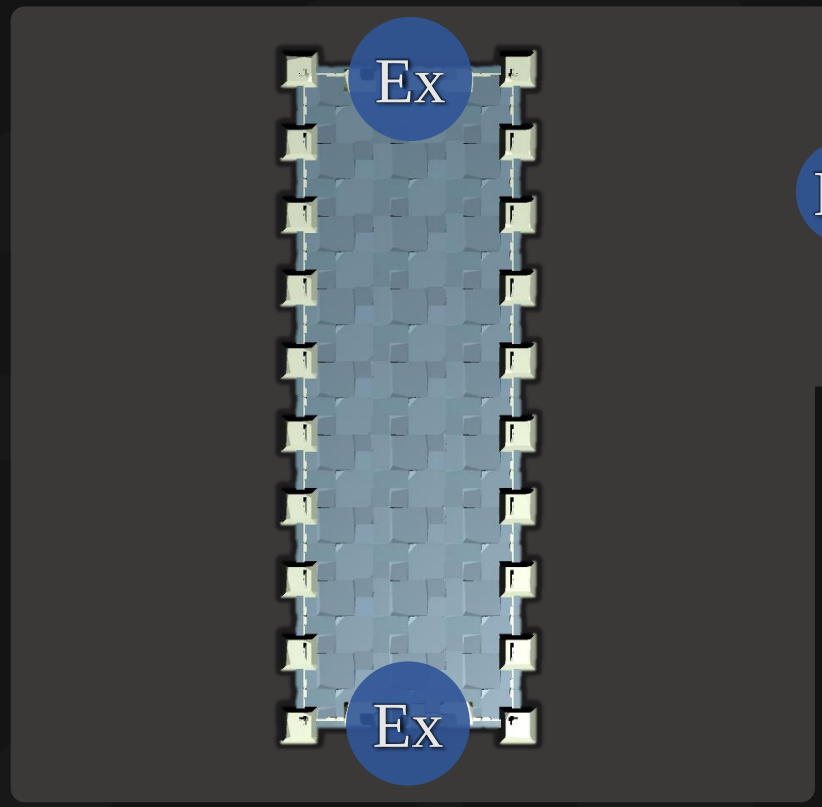


goal

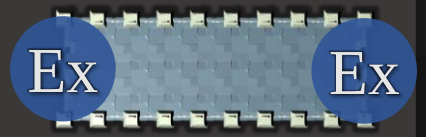


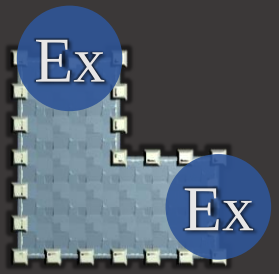
boss

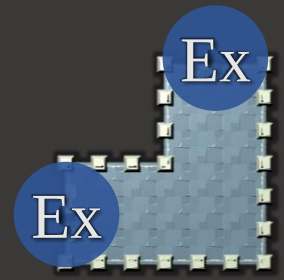
Volumes List

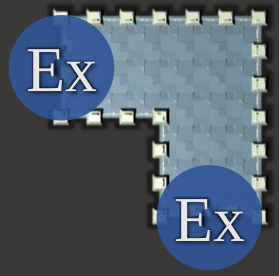


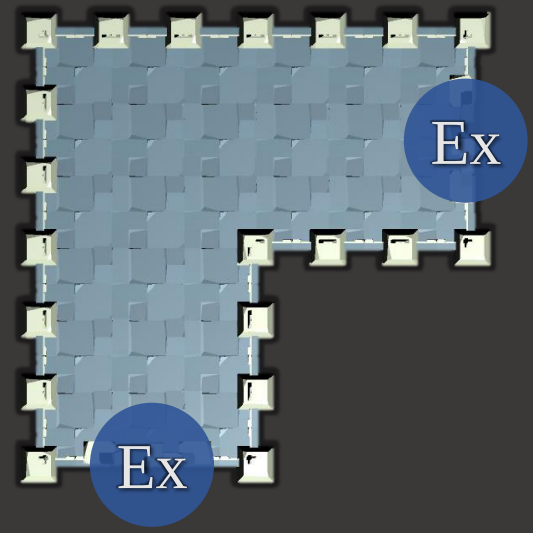
normal（I path）





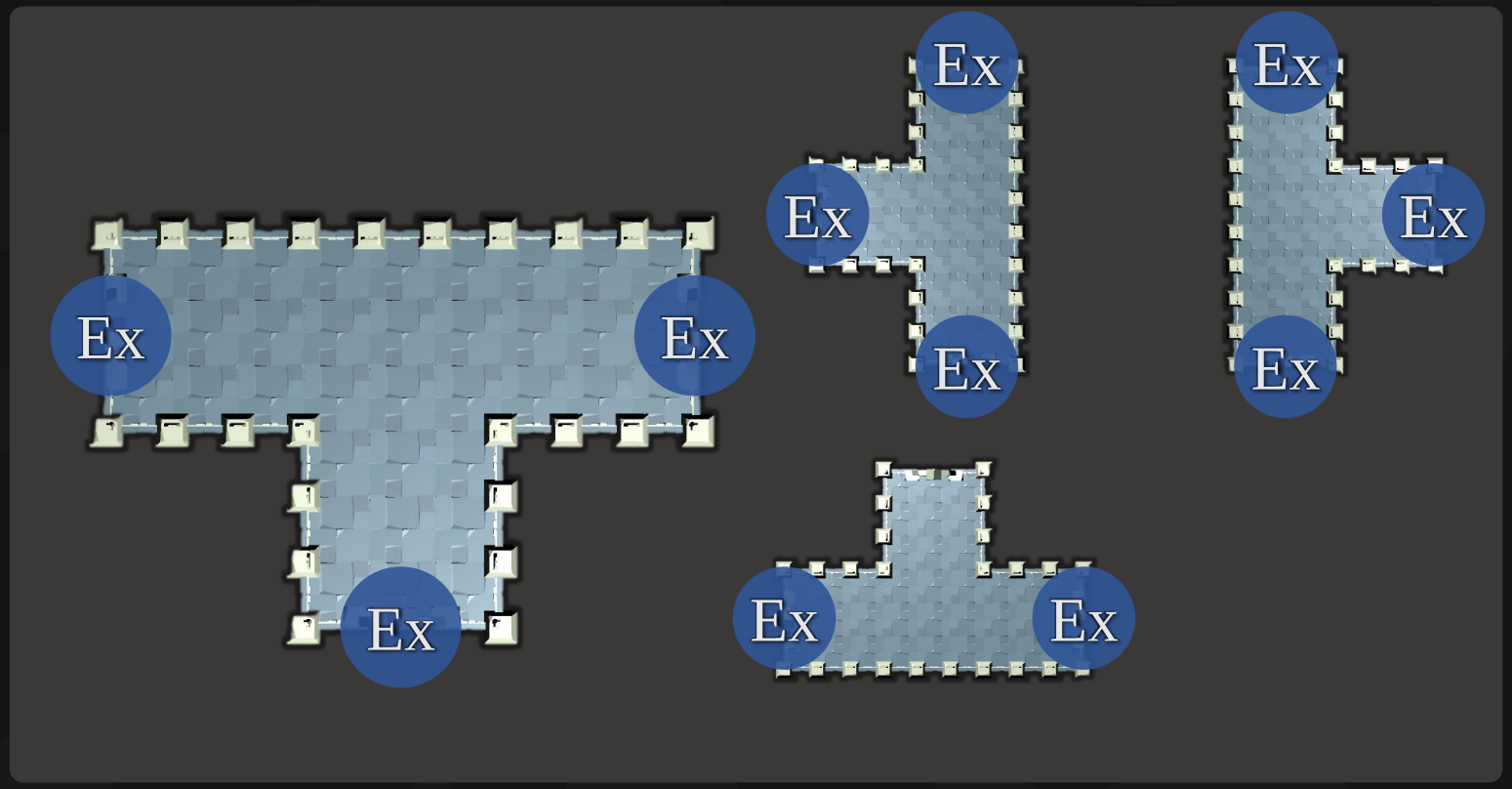




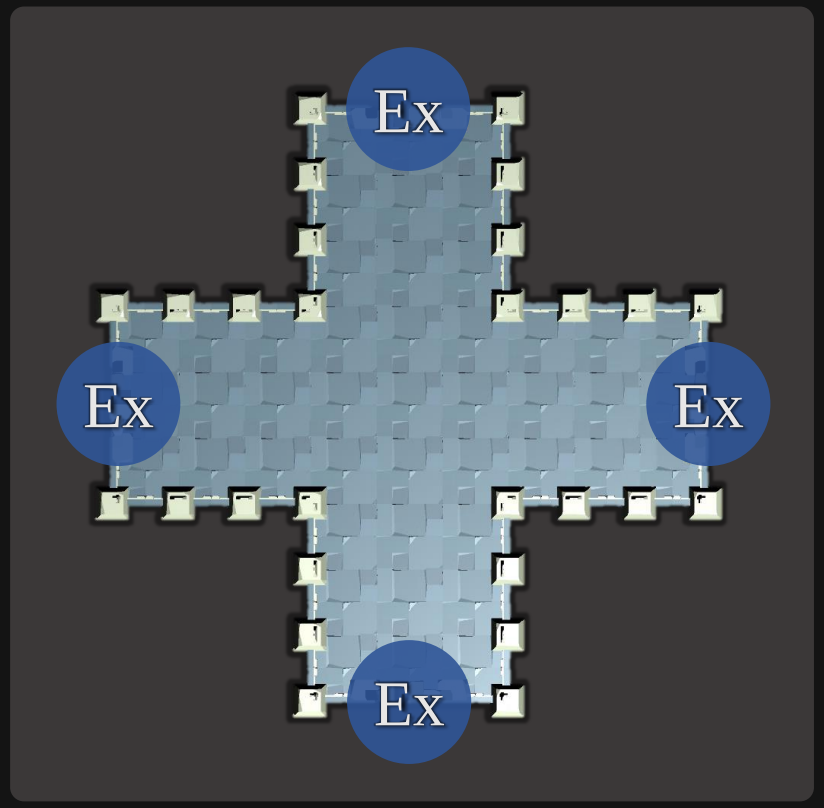


normal（L path）

Volumes List

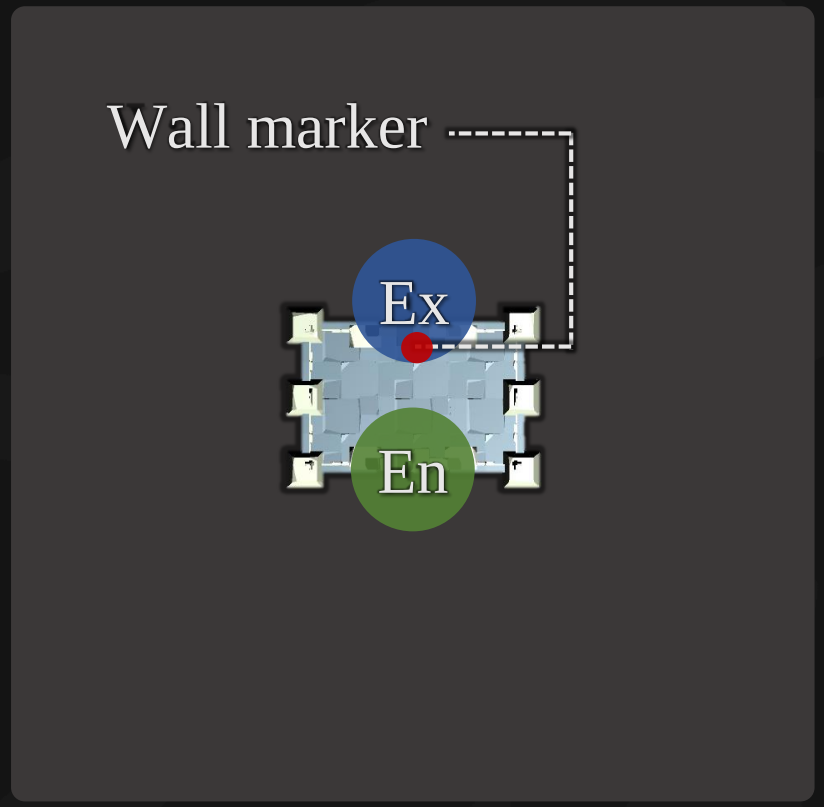


normal（T path)

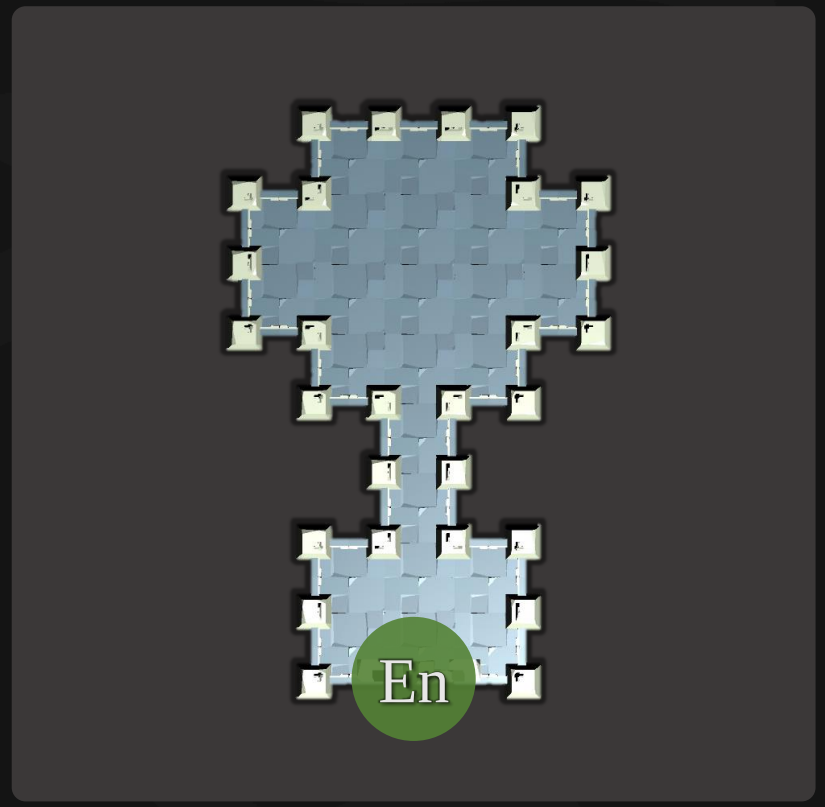


normal（+ path)

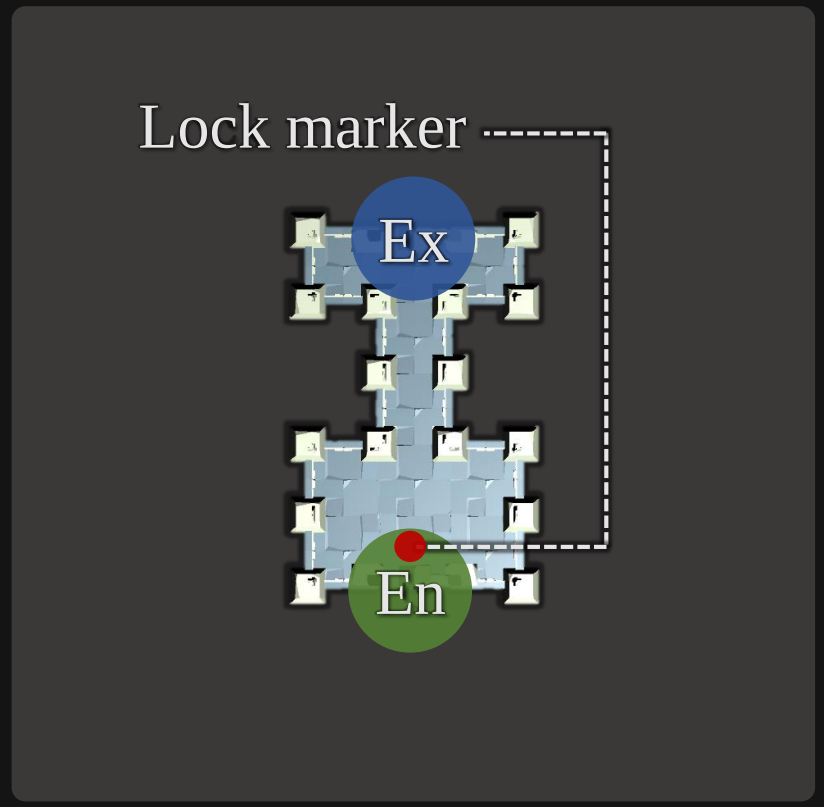
Volumes List



bomb (stable wall)

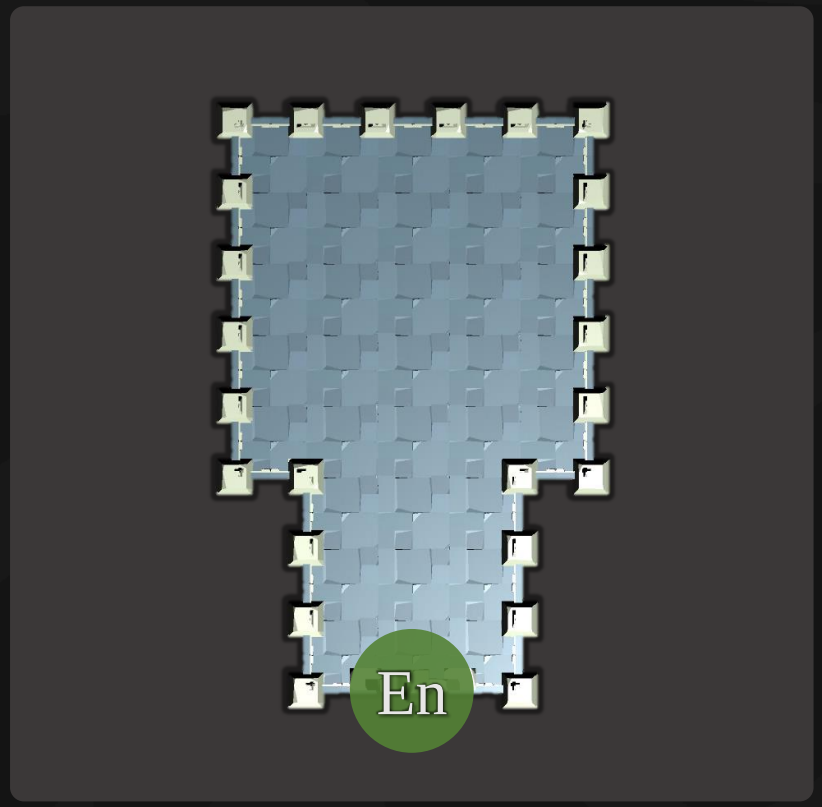


secret

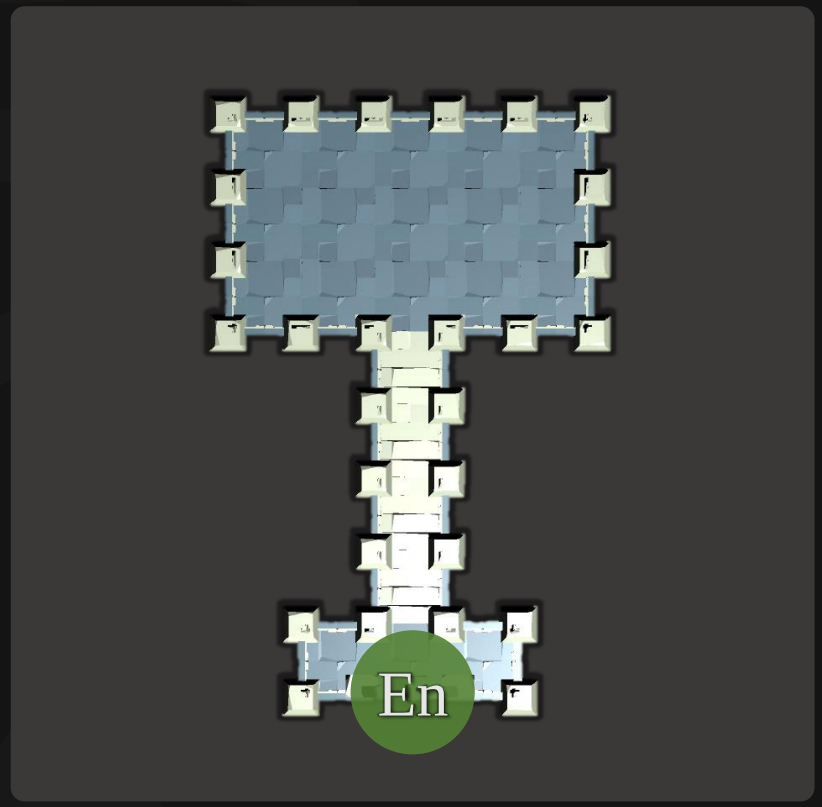


lock

Volumes List

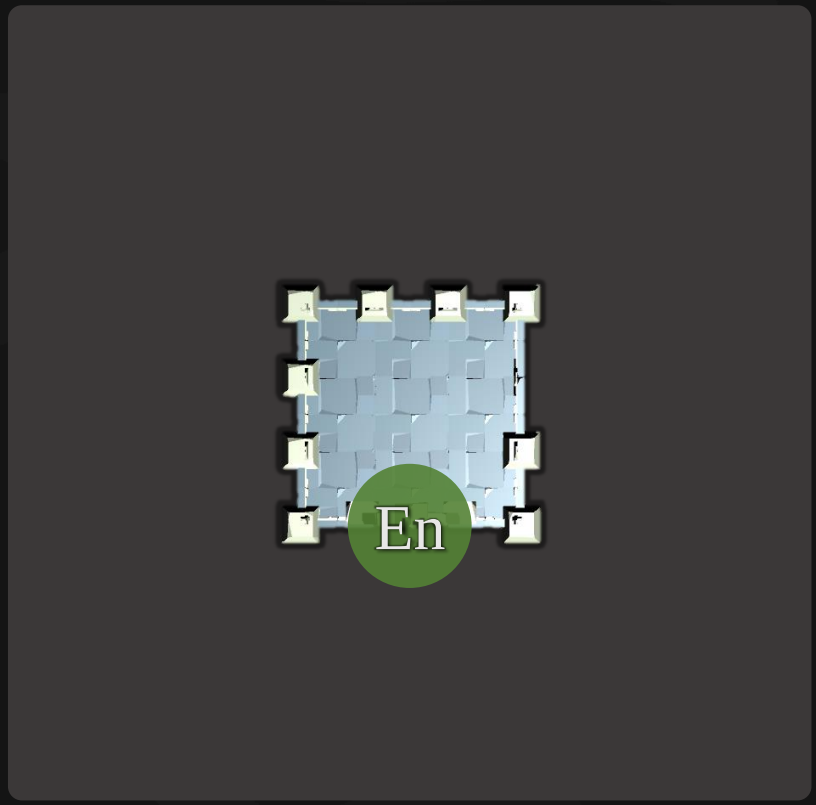


shop



treasure

Volumes List



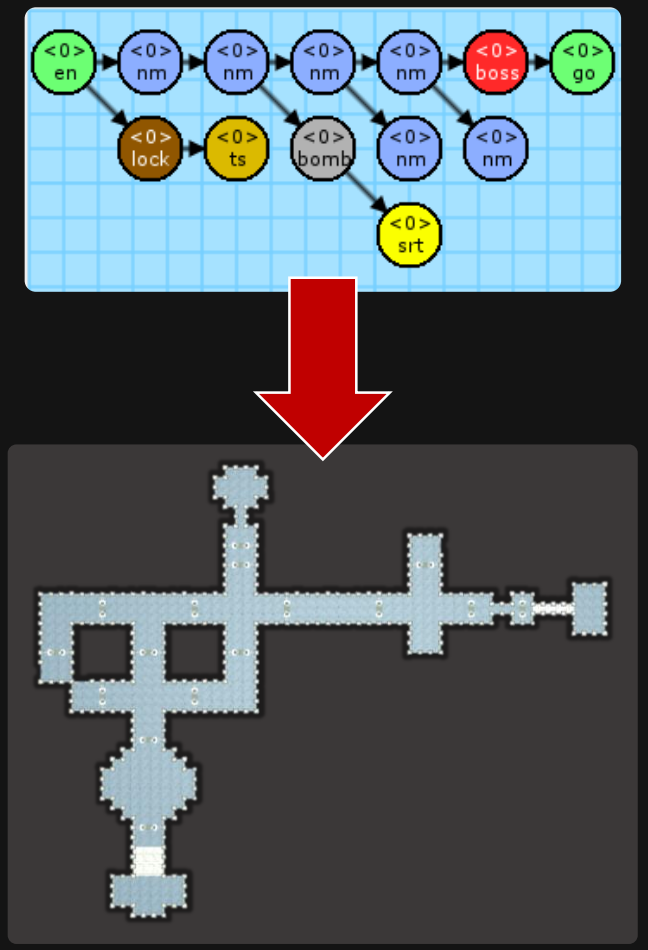
wall

Section 3. Generate Space

Mission graph transforms into level space

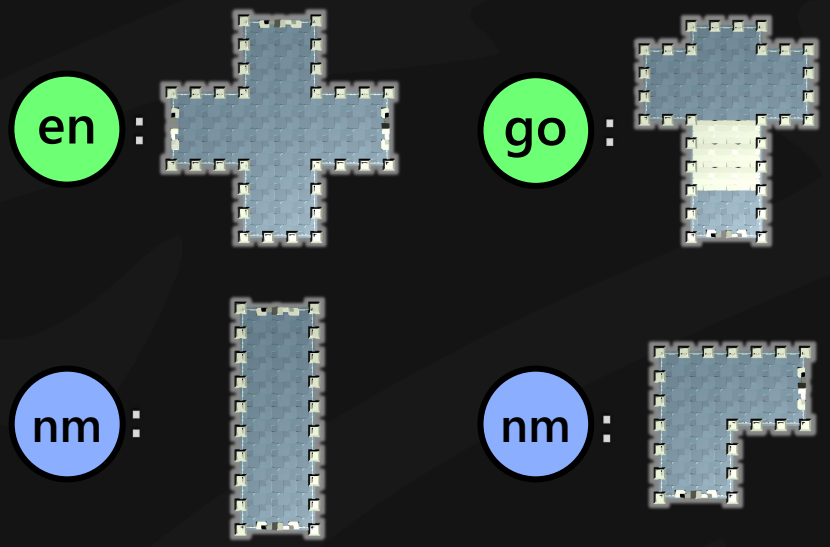
Rewrite System II

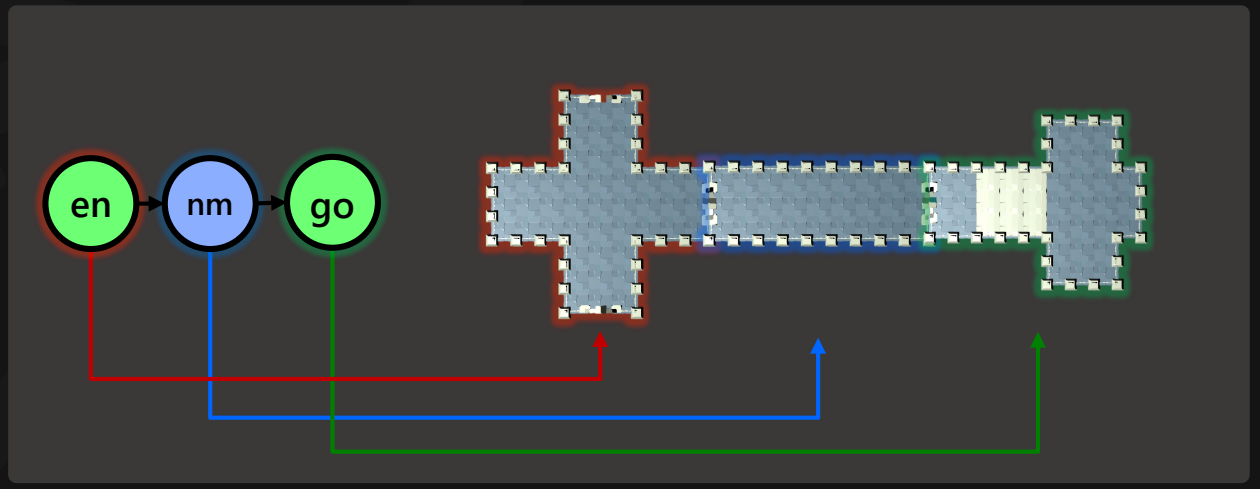
- Rewrite System II
  - Instruction
  - Replacement
  - Pseudocode
- Based on the mission graph, transform it into space



Instruction of Rewrite System II

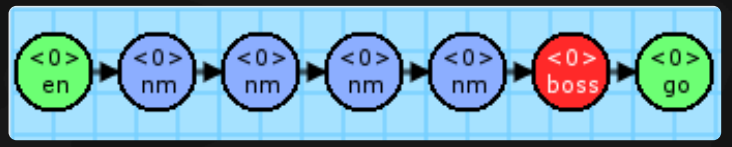
Instruction Table:

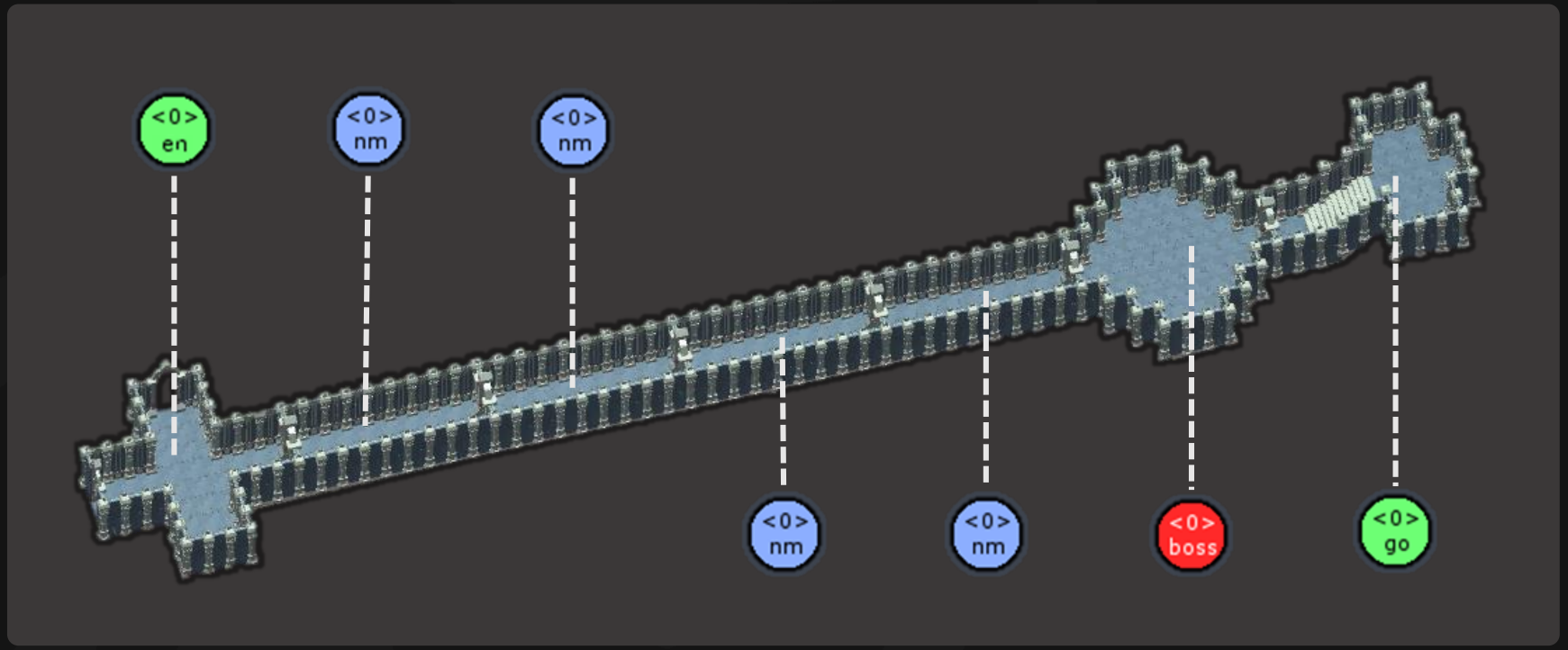




- Each rule in the shape grammar was associated with a **terminal symbol** in the mission grammar.
- Look for rules that implement that symbol, selects one at random based on their **relative weight**.

“Instruction of Rewrite System II” works



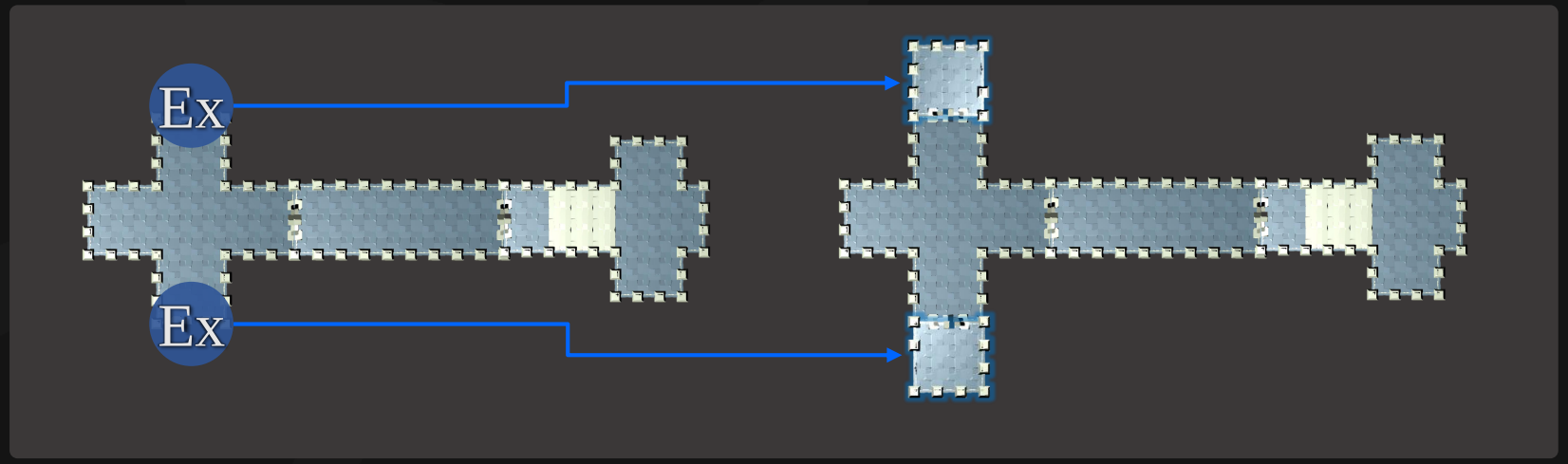


Pseudocode of Rewrite System II

```
1   Initialize random number generator via seed // Seed will affect the selection.
2   function RewriteSystem2(graph)
3       for all child ∈ graph do: // DFS
4           volume = GetVolumeFromInstructionTable(node) // Selection is based on random table.
5           isSucceed = SetVolumeToSpace(volume)         // Selection is based on random table.
6           if isSucceed is false then return false
7           end if
8       end for
...     // Another part (Replacement).
15      return true // Successfully transforms into level space.
16  end function
```

Replacement of Rewrite System II

Replacement Table:



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

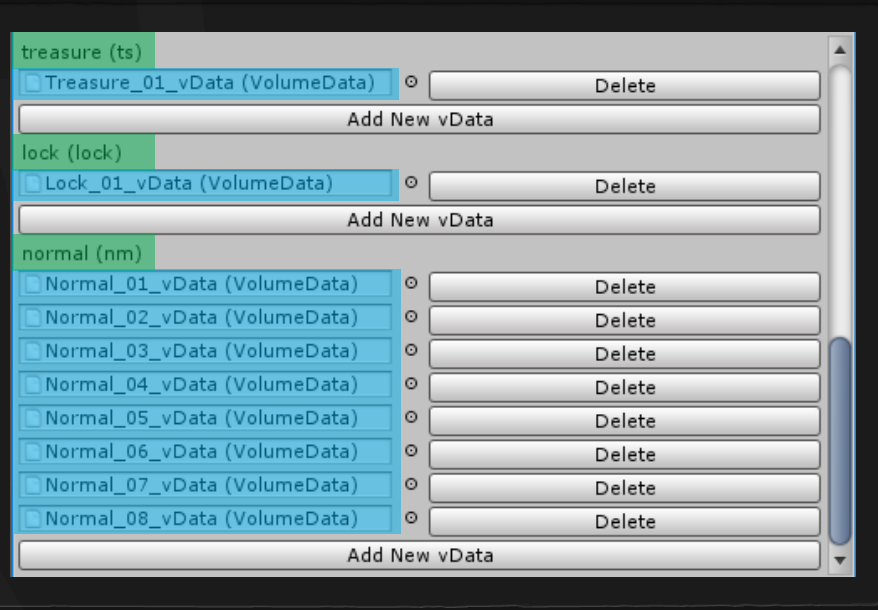
Pseudocode of Rewrite System II

```
1   Initialize random number generator via seed // Seed will affect the selection.
2   function RewriteSystem2(graph)
…       // Another part (Instruction).
9       for all marker ∈ ConnectionsInGraph do:
10          volume = GetVolumeFromReplacementTable(marker) // Selection is based on random table.
11          isSucceed = SetVolumeToSpace(volume)           // Selection is based on random table.
12          if isSucceed is false then return false
13          end if
14      end for
15      return true // Successfully transforms into level space.
16  end function
```

Manual

F Defined the Instruction of Rewrite System

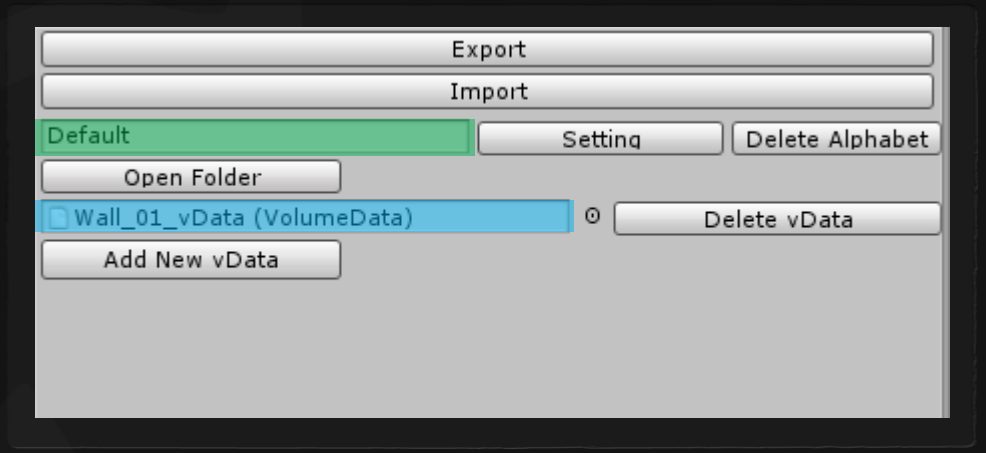
- List of the terminal symbols from Mission Grammars.
- Elements in list, they have own volume list.



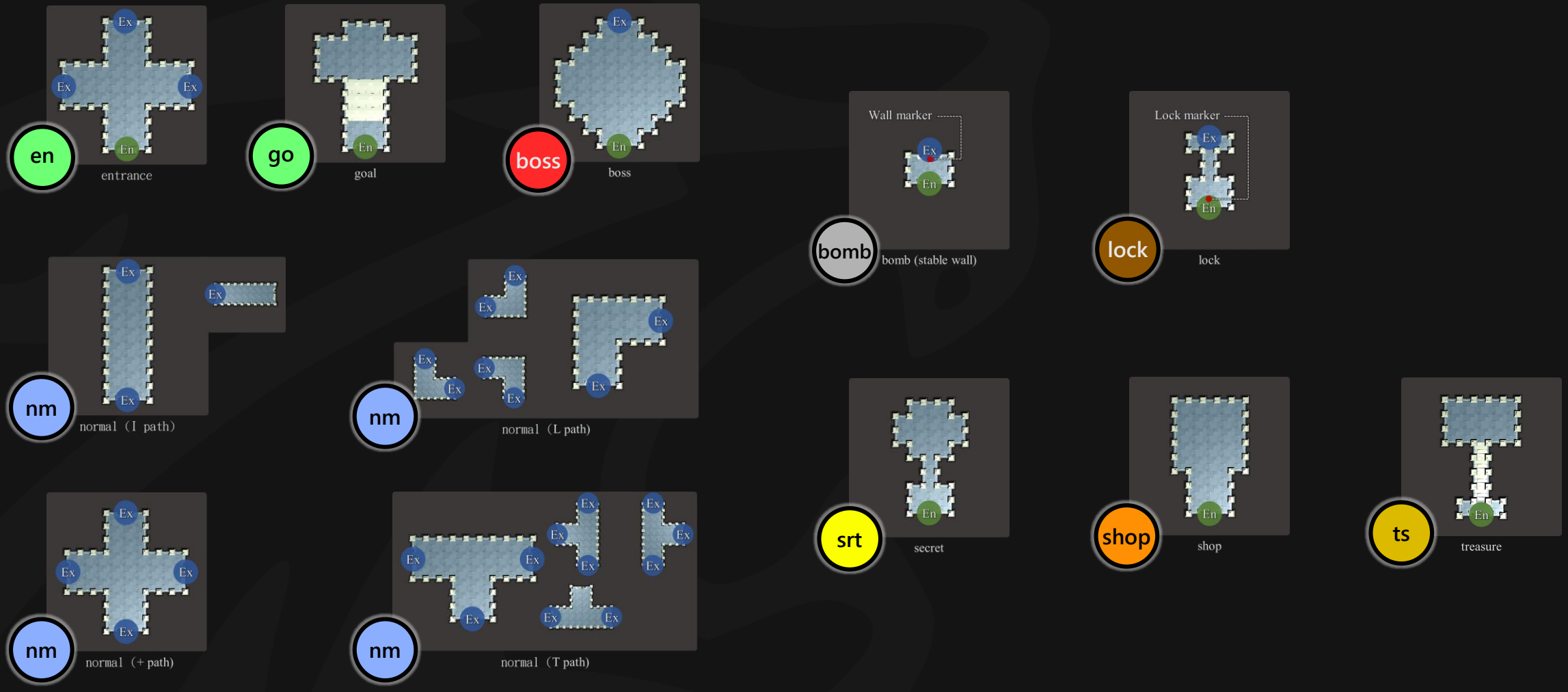
Manual

Defined the Replacement of Rewrite System

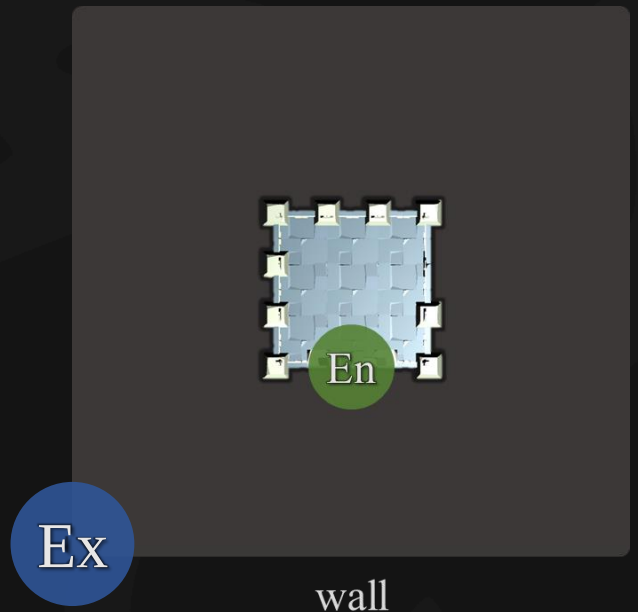
- List of the **connection markers** from Volume Manager.
- Elements in list, they have **own volume list**.



Instruction Table



Replacement Table



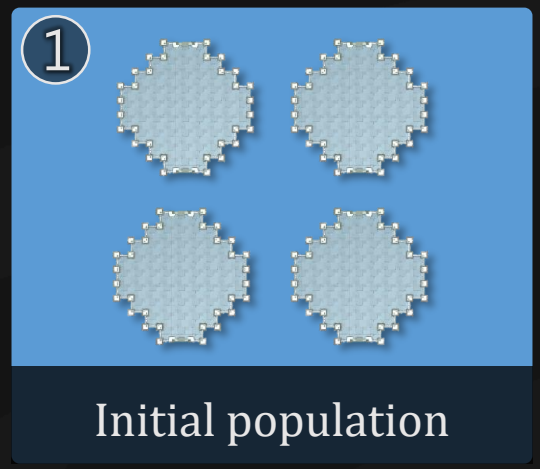
Section 4. Volume Evolution

Generate the game objects in volumes to express gameplay patterns

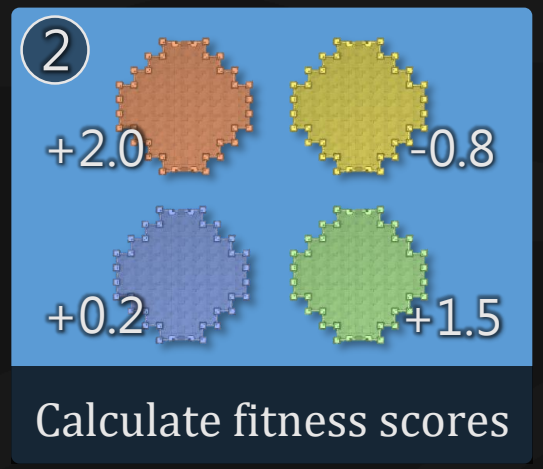
Volume Evolution

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

Algorithm flow



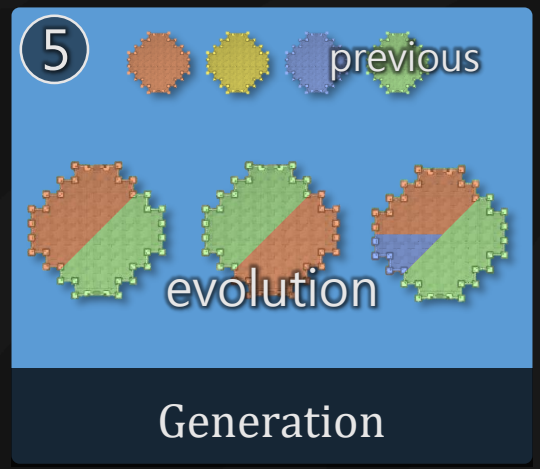
Initial population



Calculate fitness scores

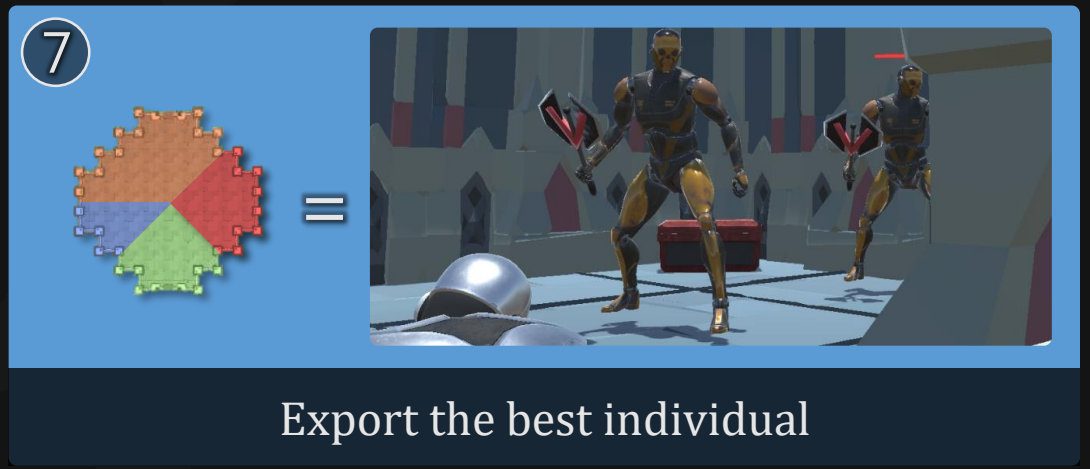
Mutation

Crossover



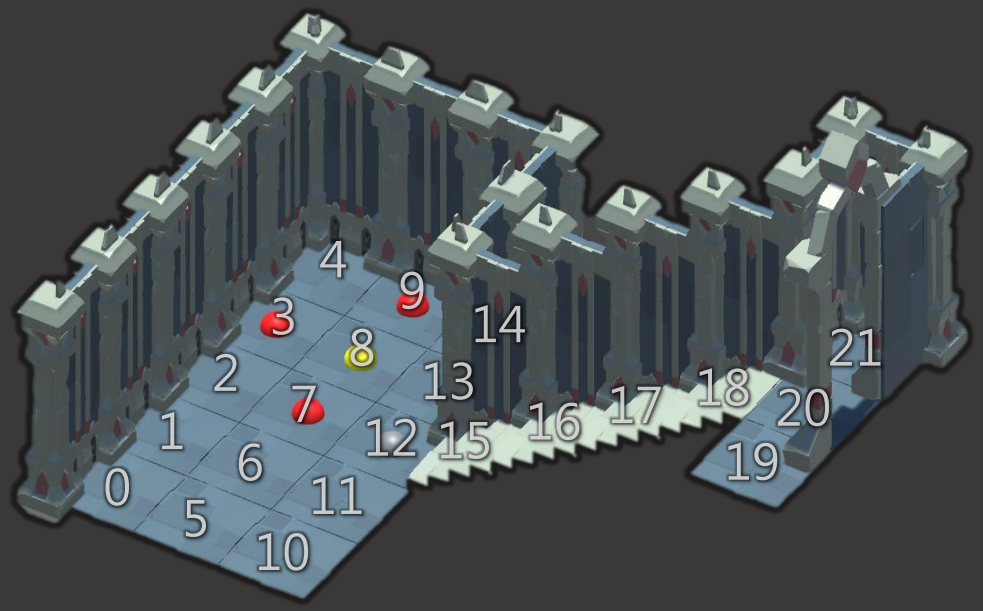
Generation

Termination condition



Export the best individual

Gene and individual



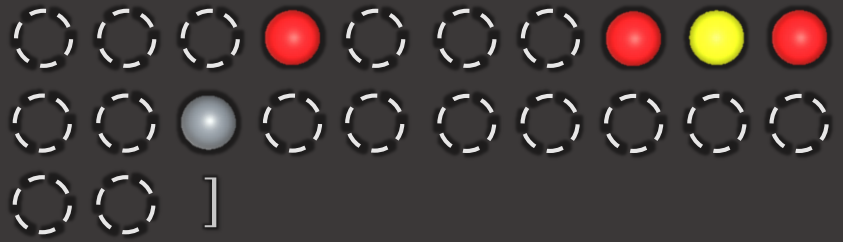
Empty

Enemy

Treasure

Trap

Chromosome of individual = [



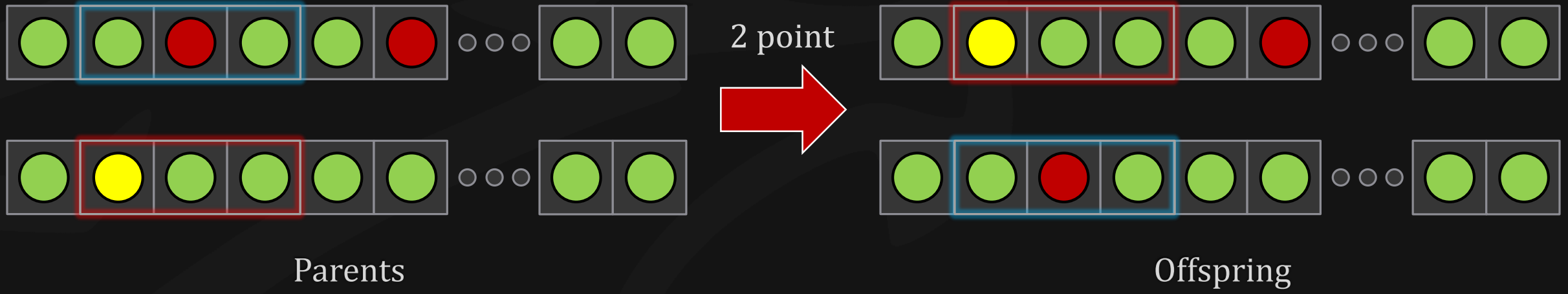
]

Gene ● = {

position: (x, y, z), type: ●

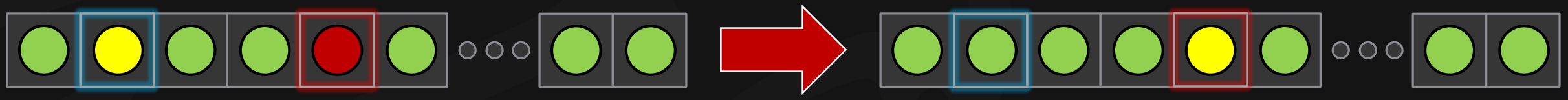
}

Crossover



- 80% crossover
- Two point crossover

Mutation



Parents

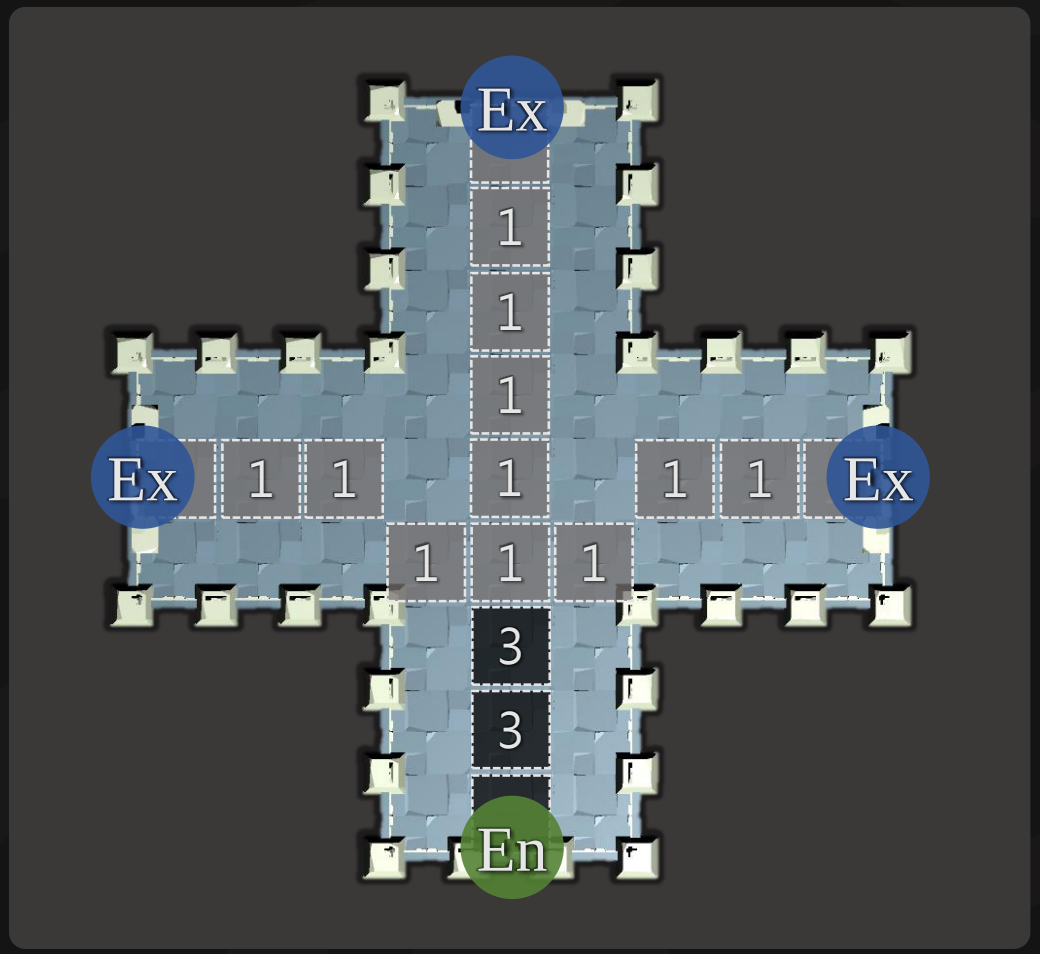
Offspring

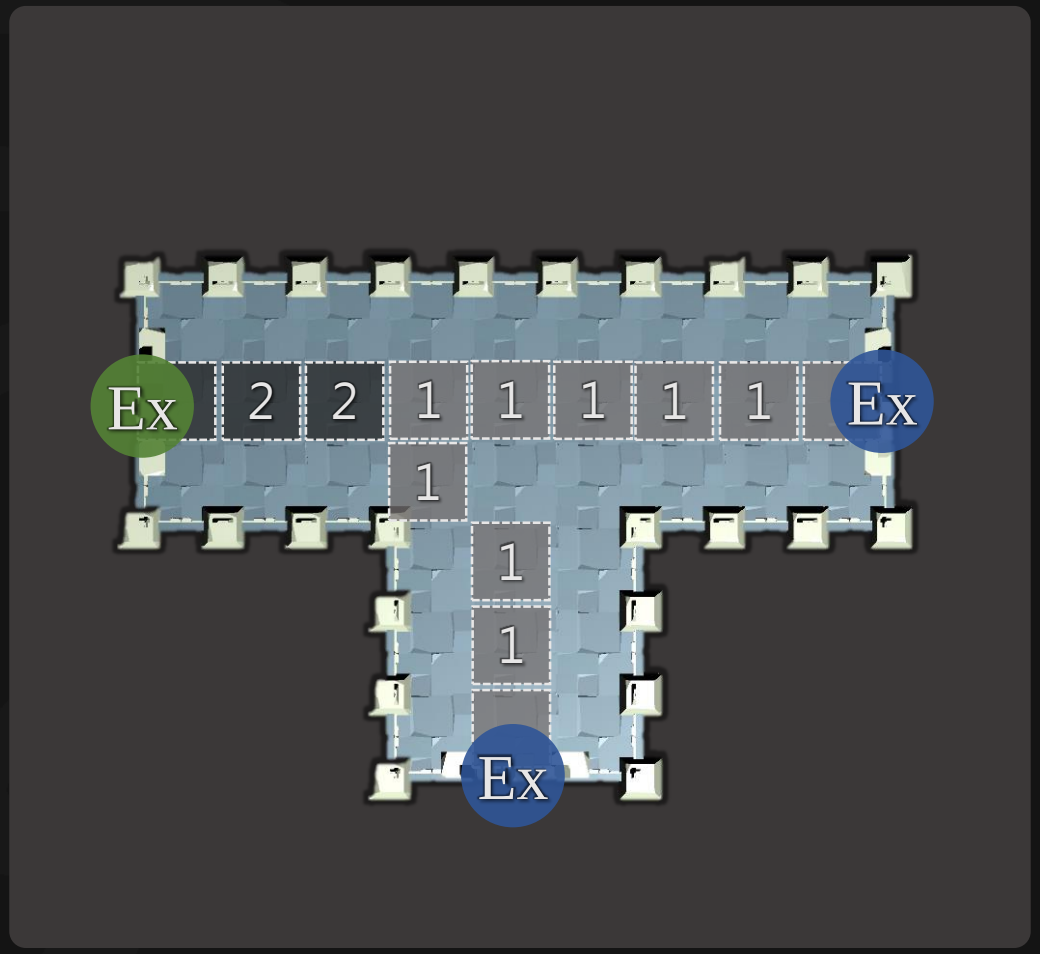
- 10% mutation
- Change 5% ~ 20% positions each chromosome

Fitness functions of GA

- Common parts
  - Main Path Calculation
  - Weight Sum Approaches
  - Normalize the fitness scores
- Metrics
  - Guard
  - Block
  - Patrol
  - Balance fitness function

Main Path Calculation using A-Star

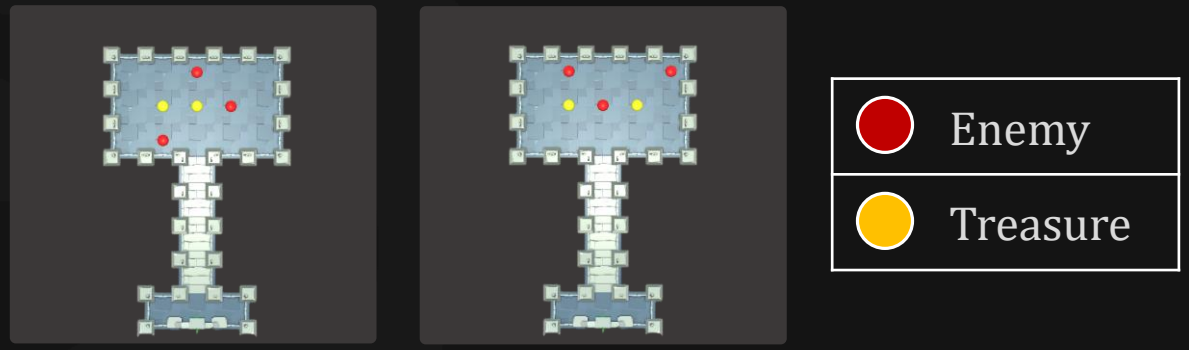




Metric: Guard

- Fitness function:

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



E_i	i-th enemy object
O_j	j-th object that be protected
$dist(a, b)$	Calculate the distance between a & b

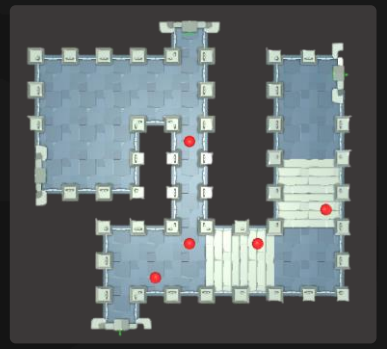
Example gameplay of "Guard"

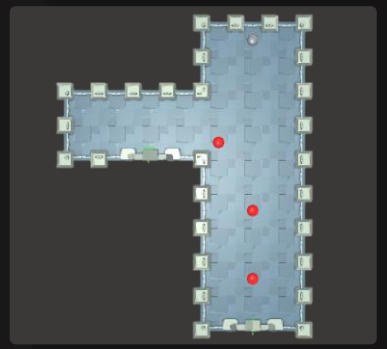
Metric: Block

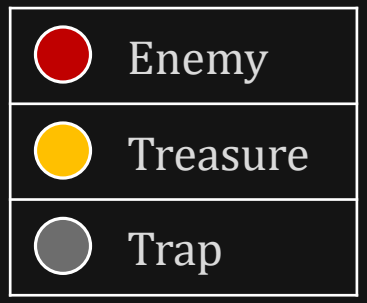
- Fitness function:

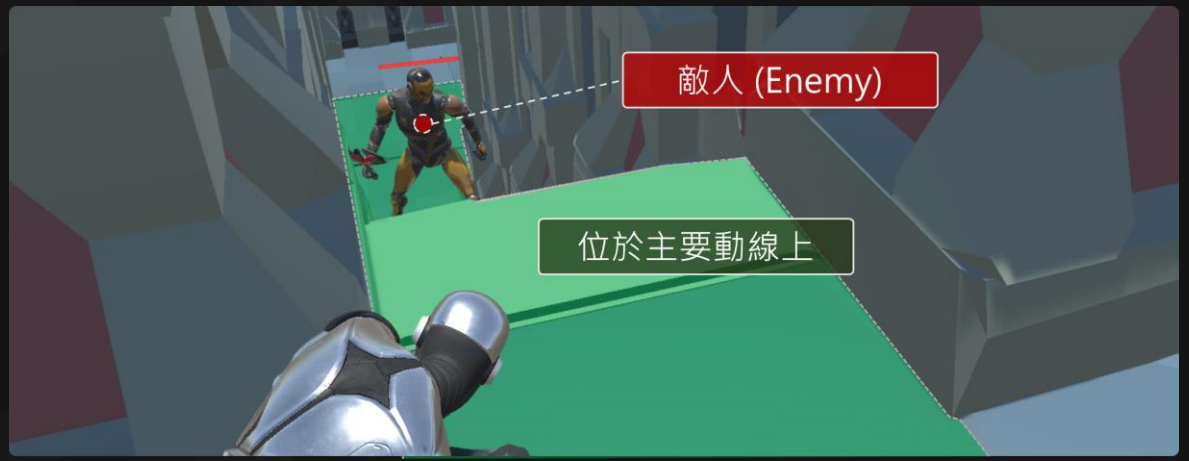
$$f_{blk} = \sum_{i=1}^{M} e_i$$

e_i	main path weight of i-th enemy object









Example gameplay of "Block"

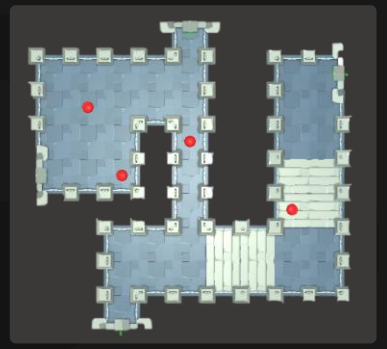
Metric: Patrol

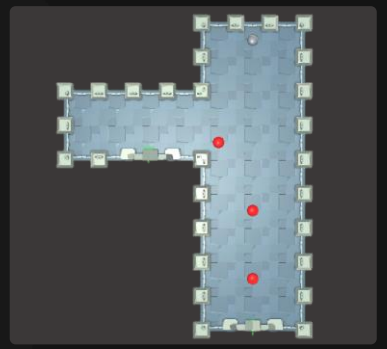
- Fitness function:

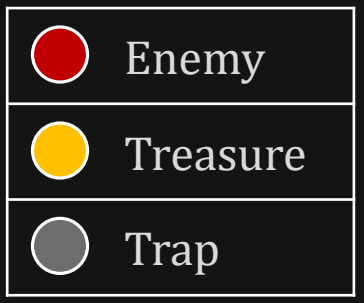
$$f_{ptl} = \sum_{i=1}^{M} neighbor(E_i, r)$$

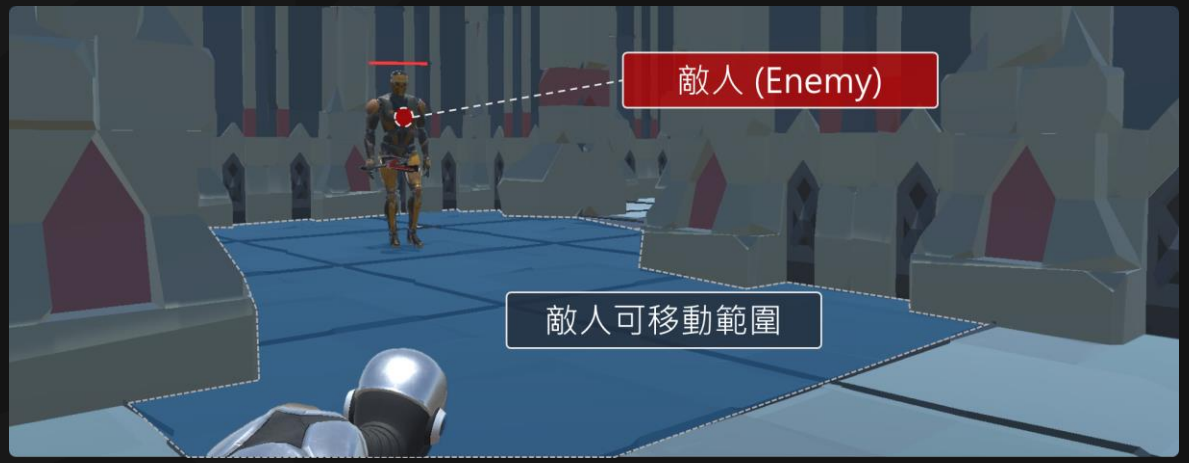
$$neighbor(Obj, r) = \sum_{j=1}^{N} isNeighbor(Obj, P_i, r) \begin{cases} 1, if\ dist(Obj, P_j) \le r \\ 0, if\ dist(Obj, P_j) > r \end{cases}$$

E_i	i-th enemy object
r	Radius of the object
P_j	j-th passable tile









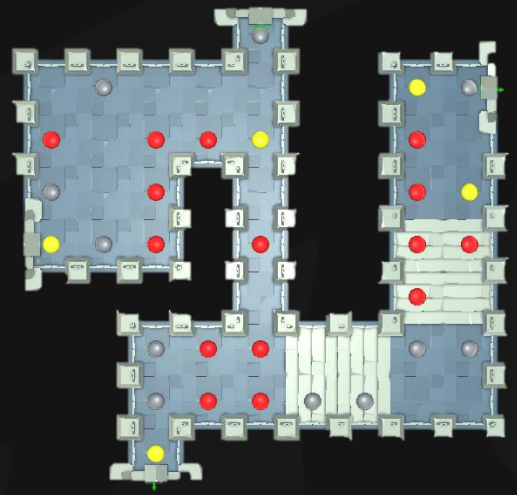
Example gameplay of "Patrol"

Balance fitness function: Density

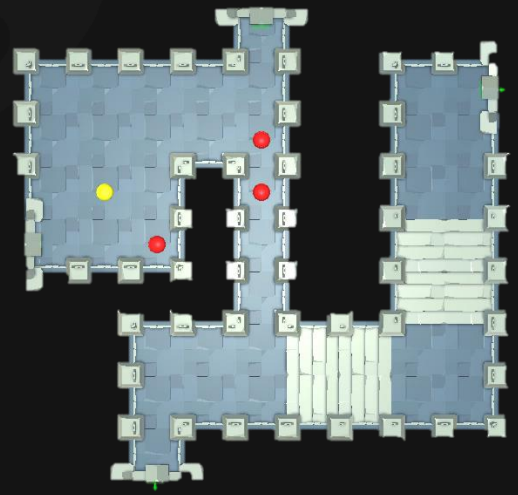
- Density fitness function:

$$f_{dst} = \begin{cases} 1 \text{ , } N \leq O \leq M \\ 0 \text{ , } if\ O < N\ or\ O > N \end{cases}$$

M	M is upper bound of gameobject
N	N is lower bound of gameobject
O	O is current number of gameobject



$score = 0$



$score = 1$

Weight Sum Approaches

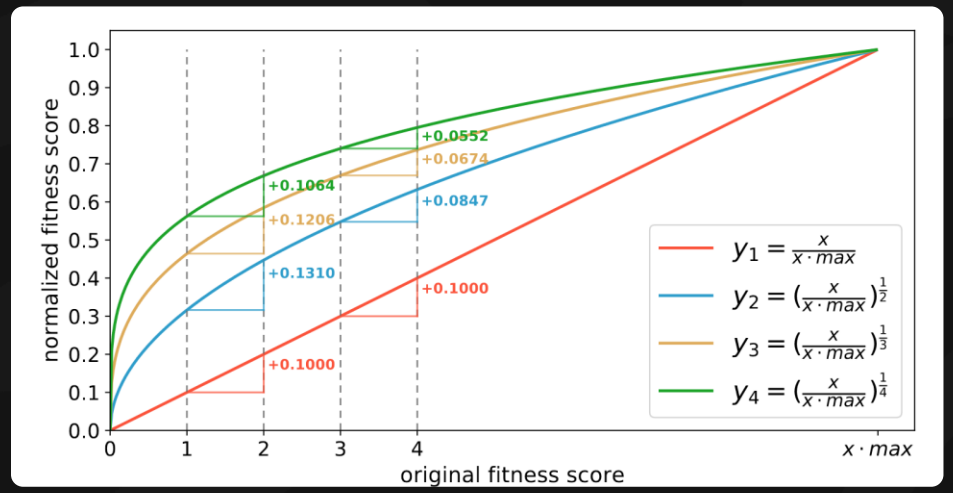
- Weight sum approaches:

$$f_{all} = \sum_{i=1}^{M} (Normalized(f_i) \times \omega_i)$$

$$Normalized(f_i) = \left(\frac{f_i}{f_i \cdot max} \right)^{\frac{1}{c}}$$

M	M types fitness functions
f_i	i-th fitness score
ω_i	Weight of the i-th fitness score
$f_i \cdot max$	Maximum of all i-th fitness scores

Normalize the fitness scores



- $y_1 = f_1(x)$ is the original normalization
- $y_2 = f_2(x)$ is we expected normalization

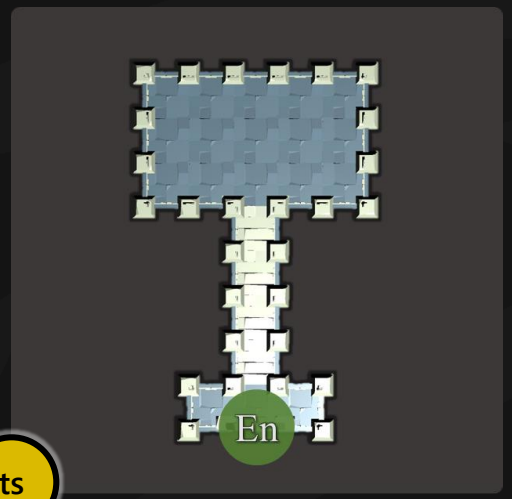
- Parameters
  - x : Un-normalized score
  - $x \cdot max$: Maximum of un-normalized score
  - $f(x)$: Normalized score
- An example. If $N = 10$
  - $f_1(0) = 0.0000$; $f_2(0) = 0.0000$
    $+ 0.1000$ $+ 0.3162$
  - $f_1(1) \cong 0.1000$; $f_2(1) = 0.3162$
    $+ 0.1000$ $+ 0.1310$
  - $f_1(2) \cong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

Manual

H Set the weights of fitness functions

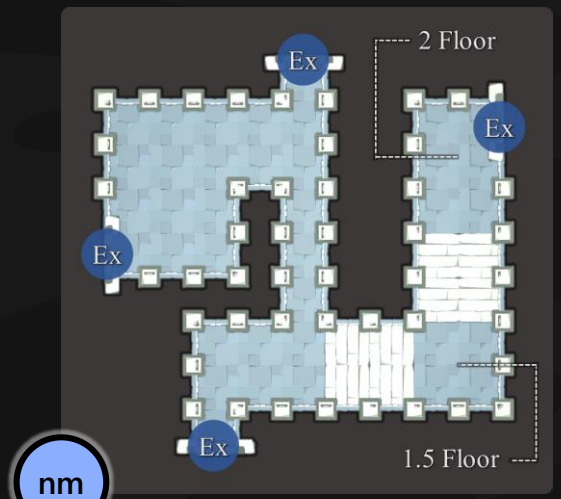
- Setting about fitness function weights of each volume
- Same volumes can be duplicated used on different positions

Treasure and appended volumes



treasure

w_{guard}	1.00	
$w_{density}$	1.00	[3, 5]



normal (narrow path)

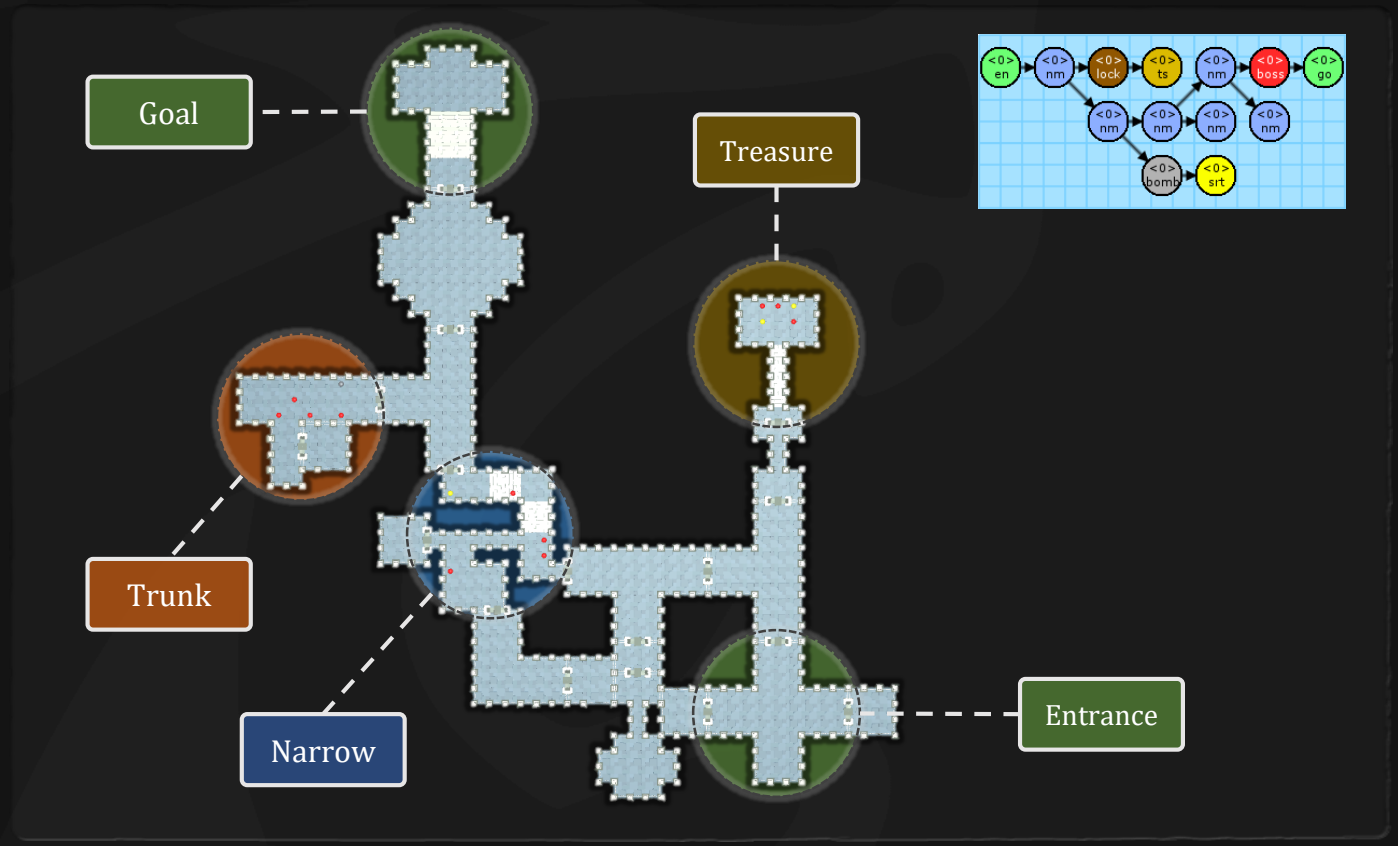
w_{block}	1.00	
w_{patrol}	0.75	
$w_{density}$	1.75	[4, 5]

normal (trunk path)

w_{block}	1.00	
w_{patrol}	0.50	
w_{guard}	-1.00	
$w_{density}$	2.50	[3, 5]

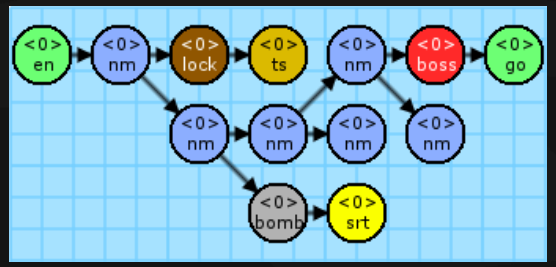
Manual

I Export the completed level

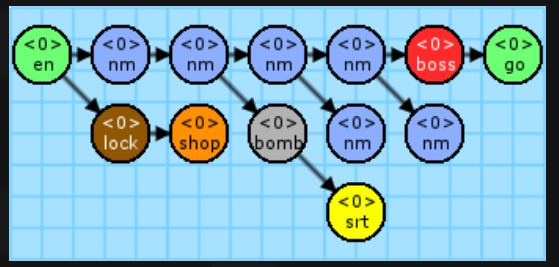


Experimental Results

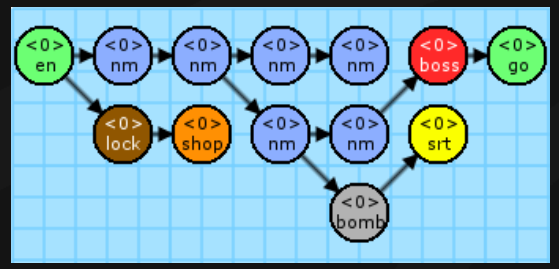
Generate the Mission Graph



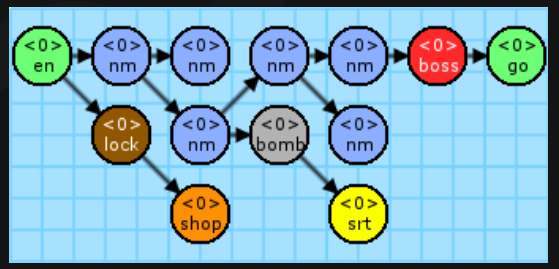
(a) Seed = 727427



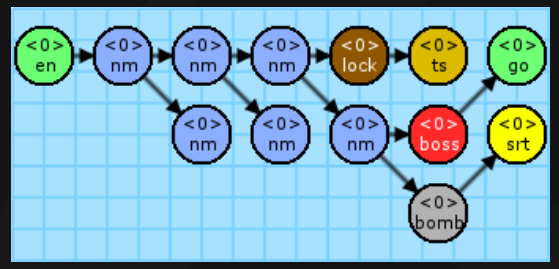
(b) Seed = 785428



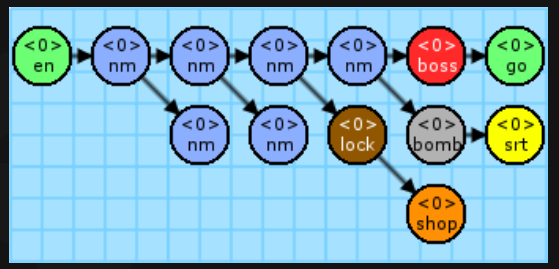
(c) Seed = 731269



(d) Seed = 236909

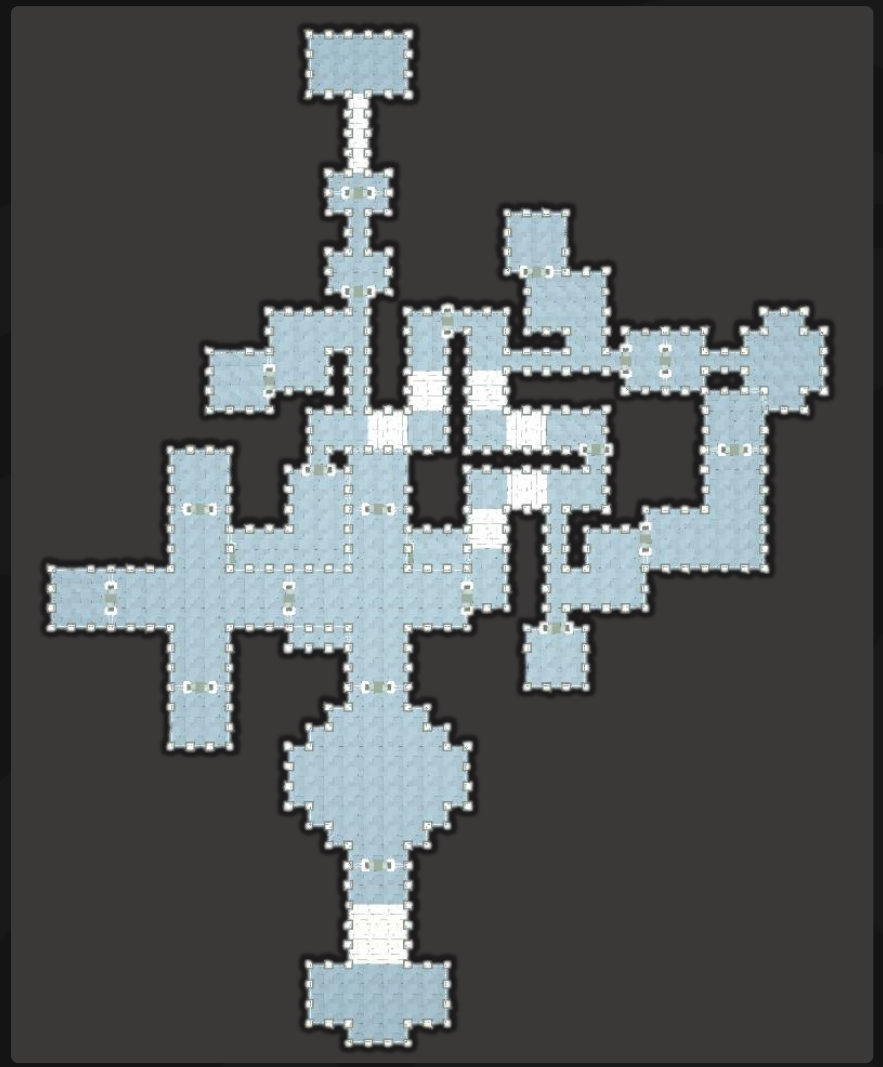


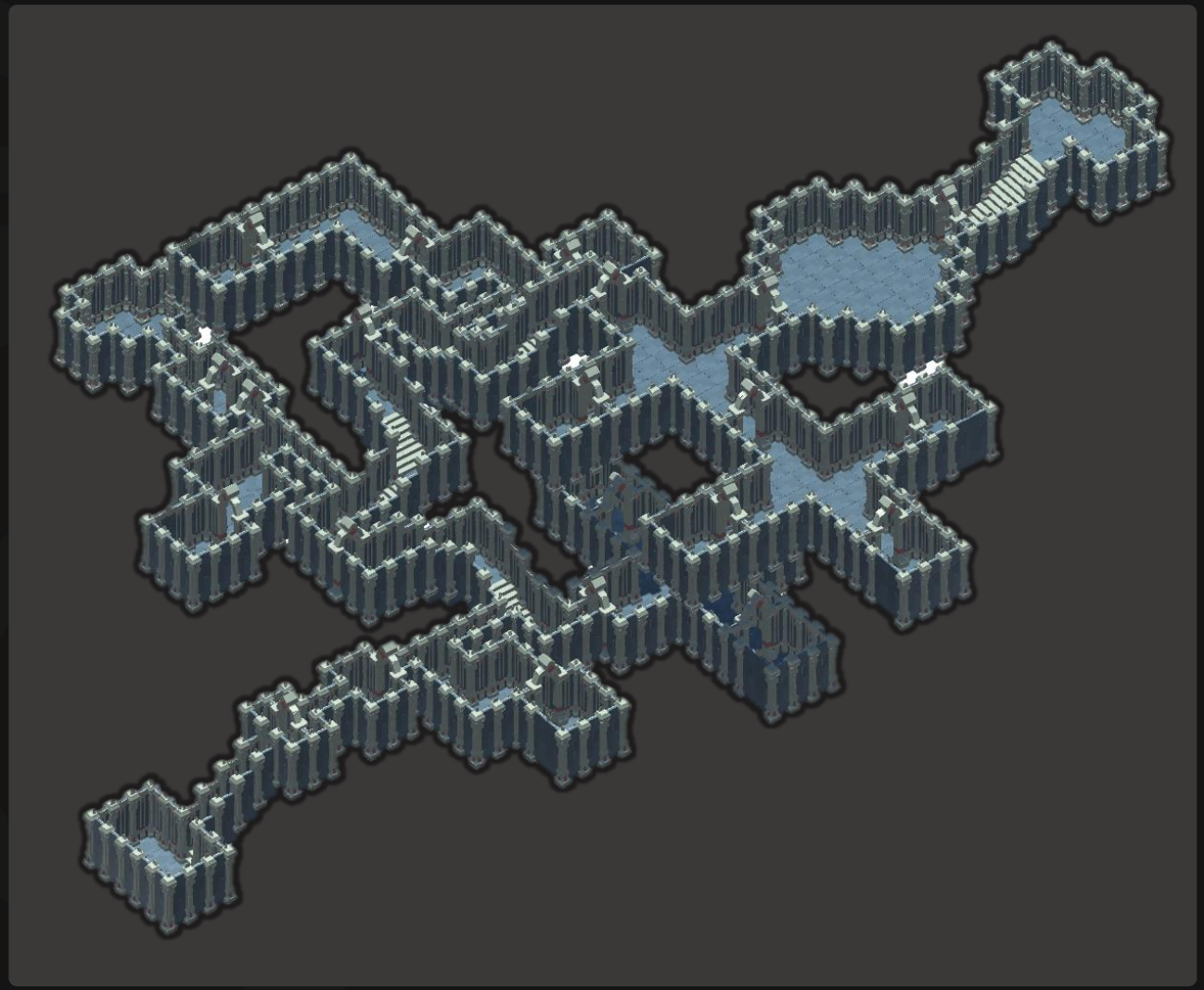
(e) Seed = 514113



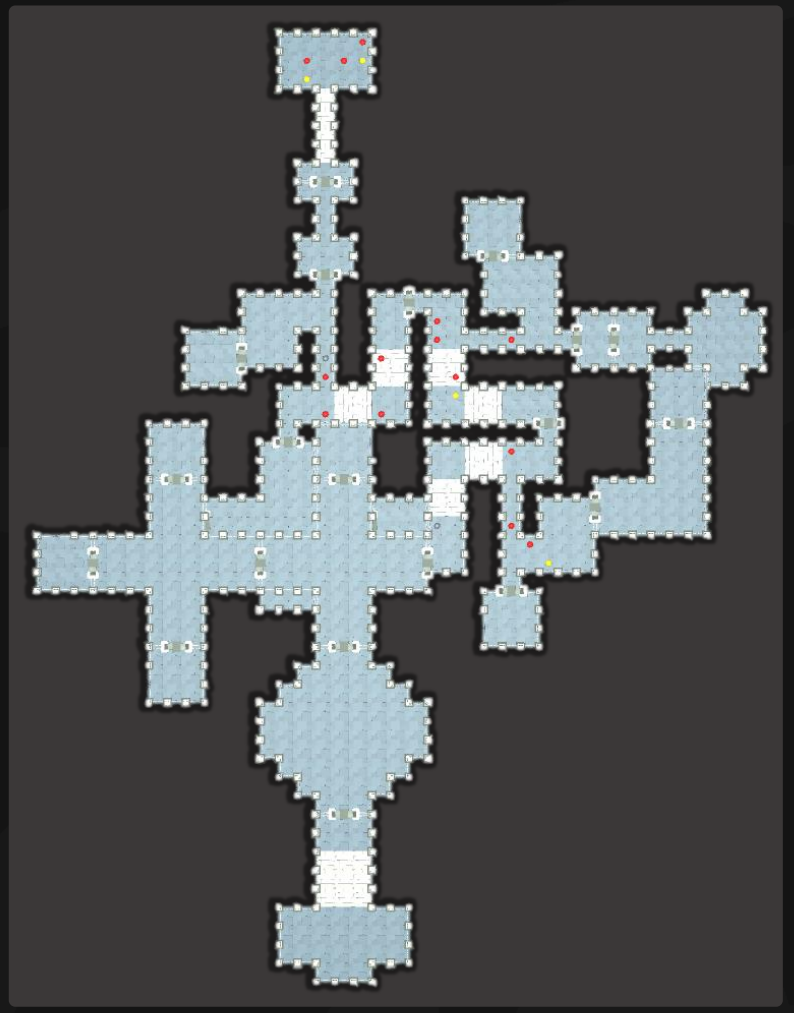
(f) Seed = 472184

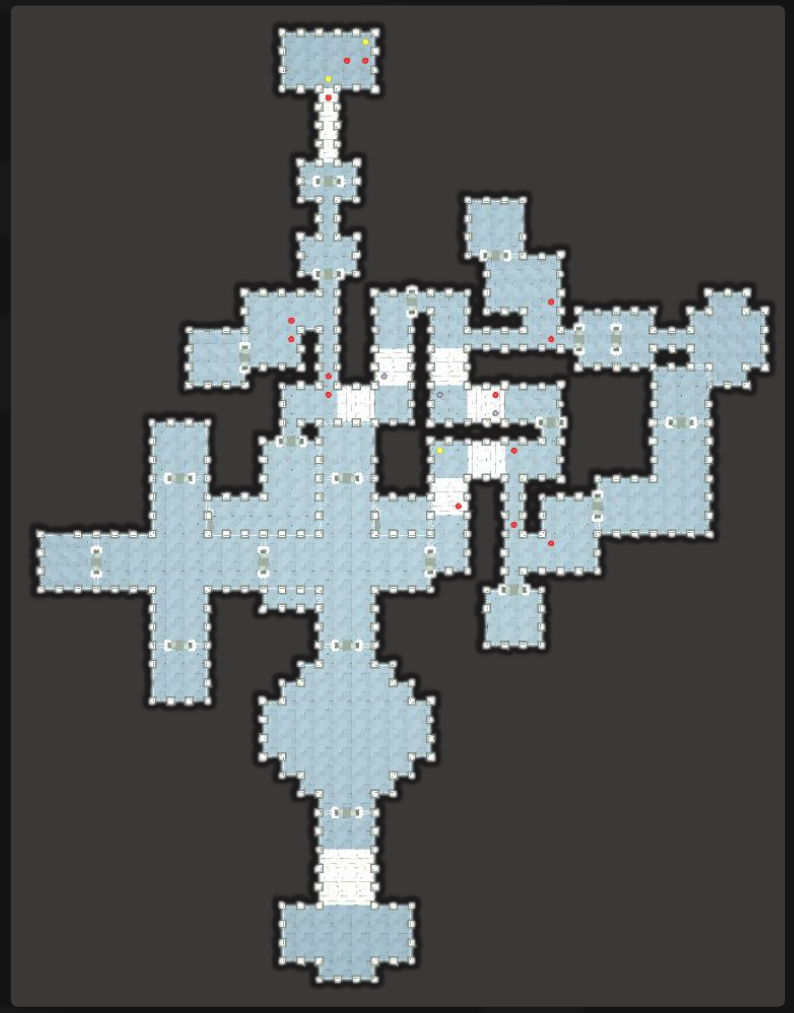
Transform into Level Space (A)

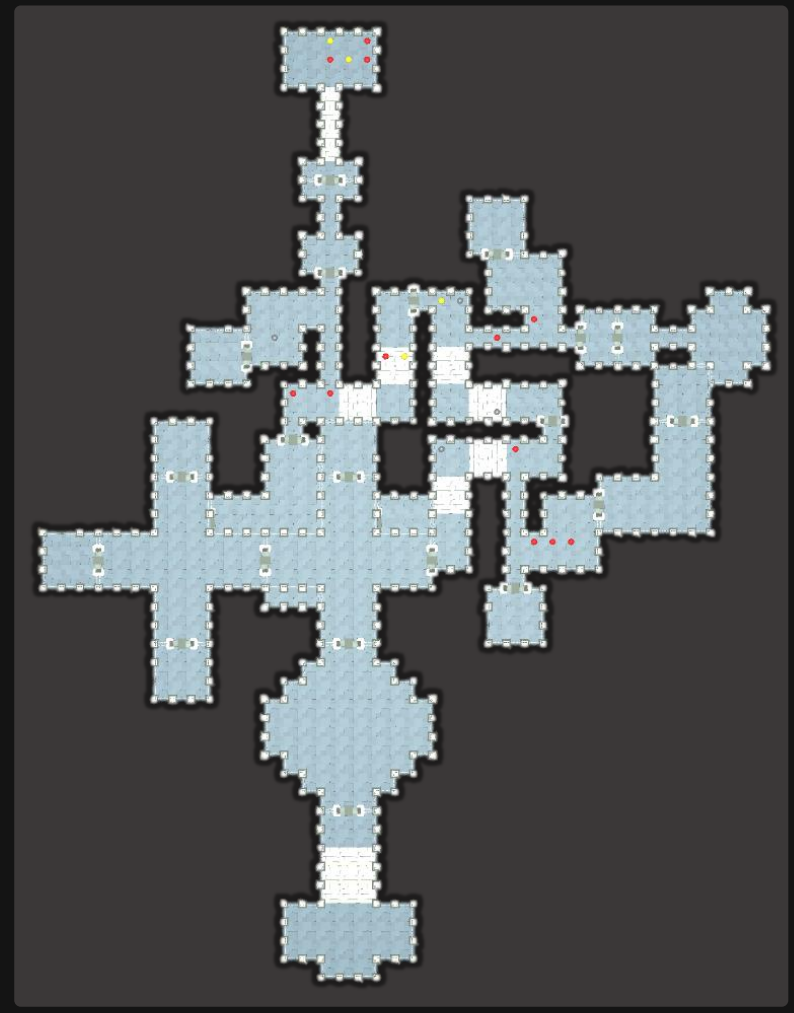




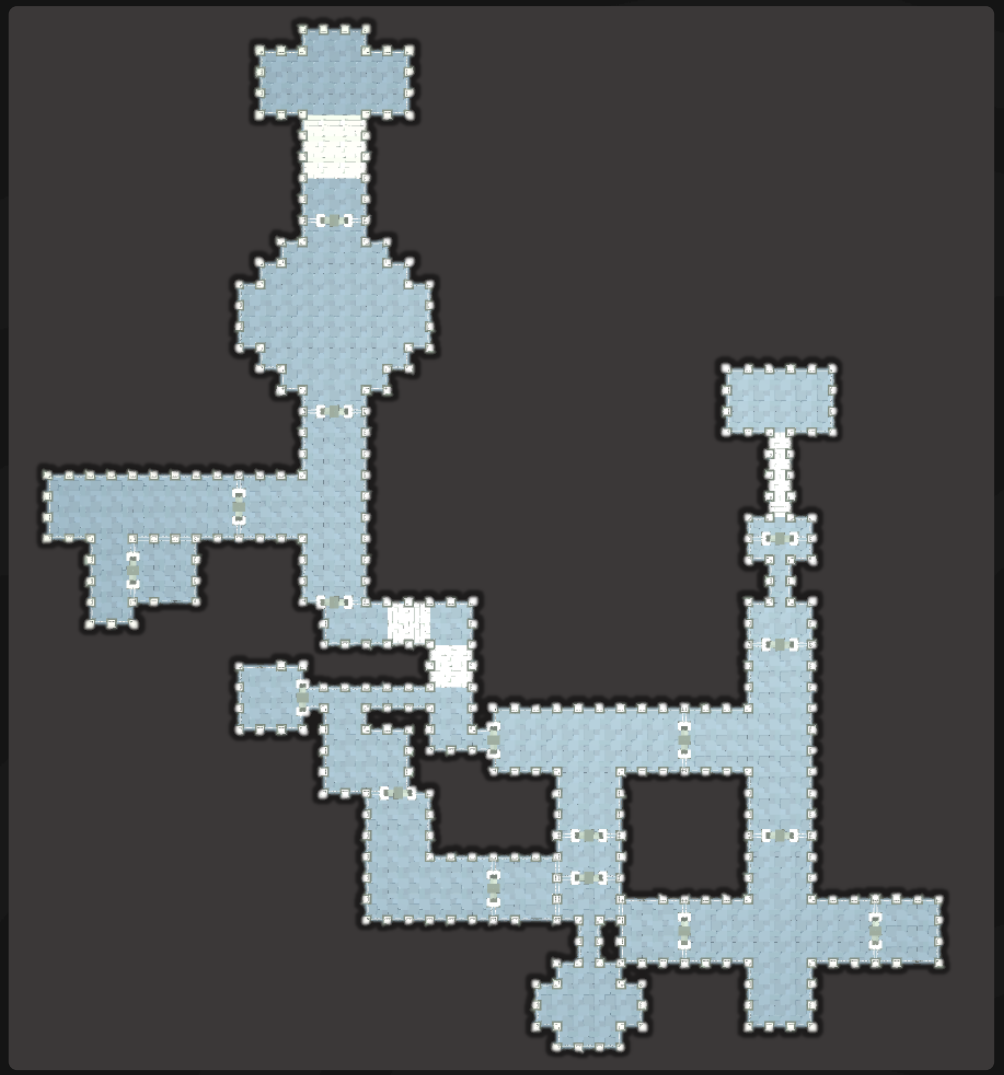
Generate the game objects (A)

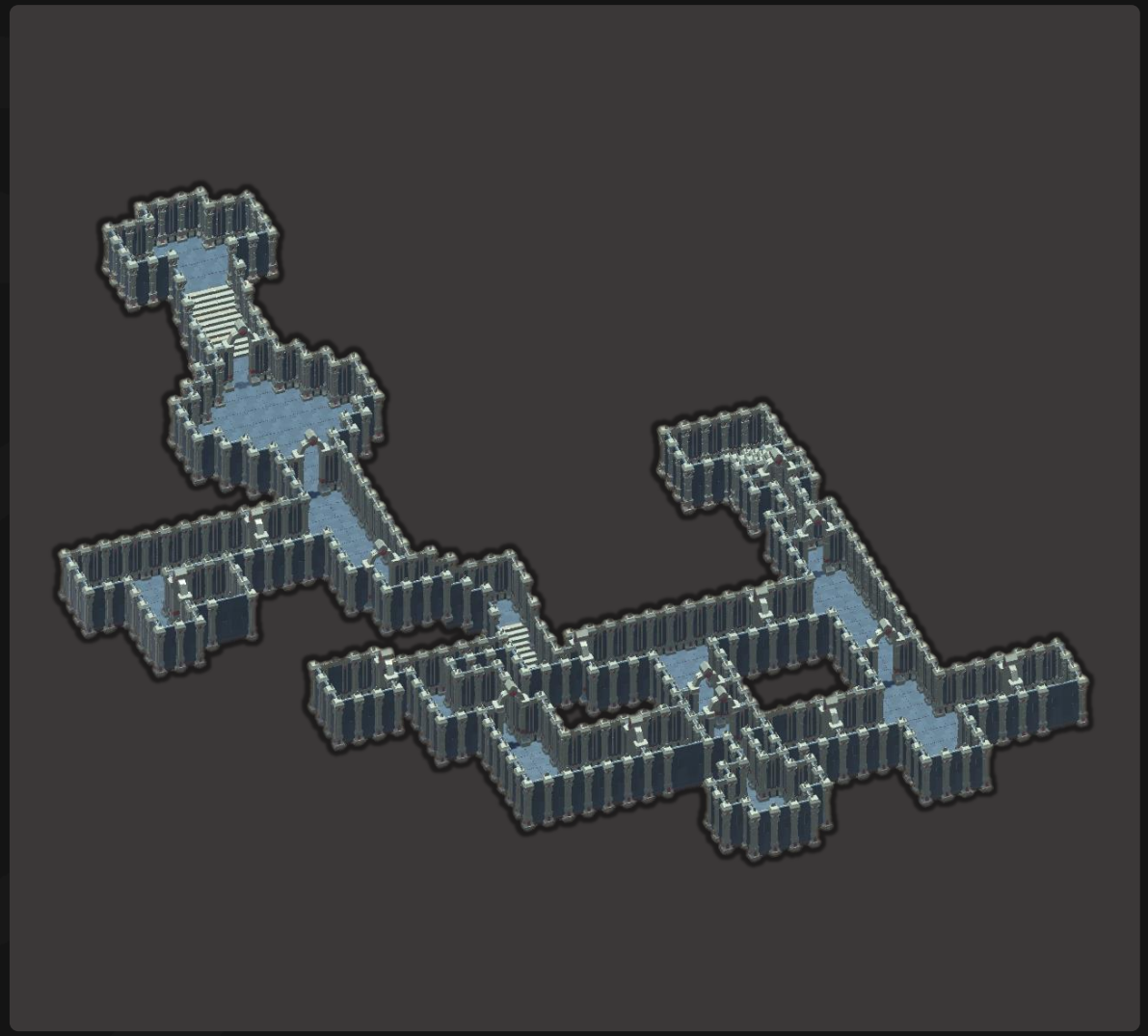




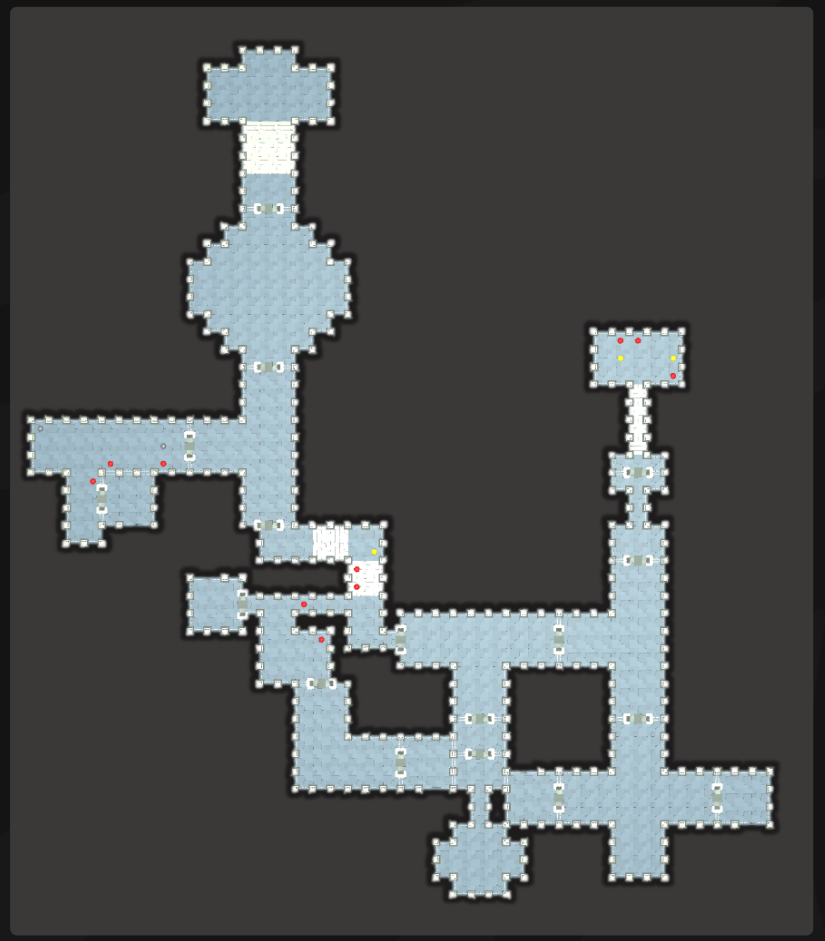


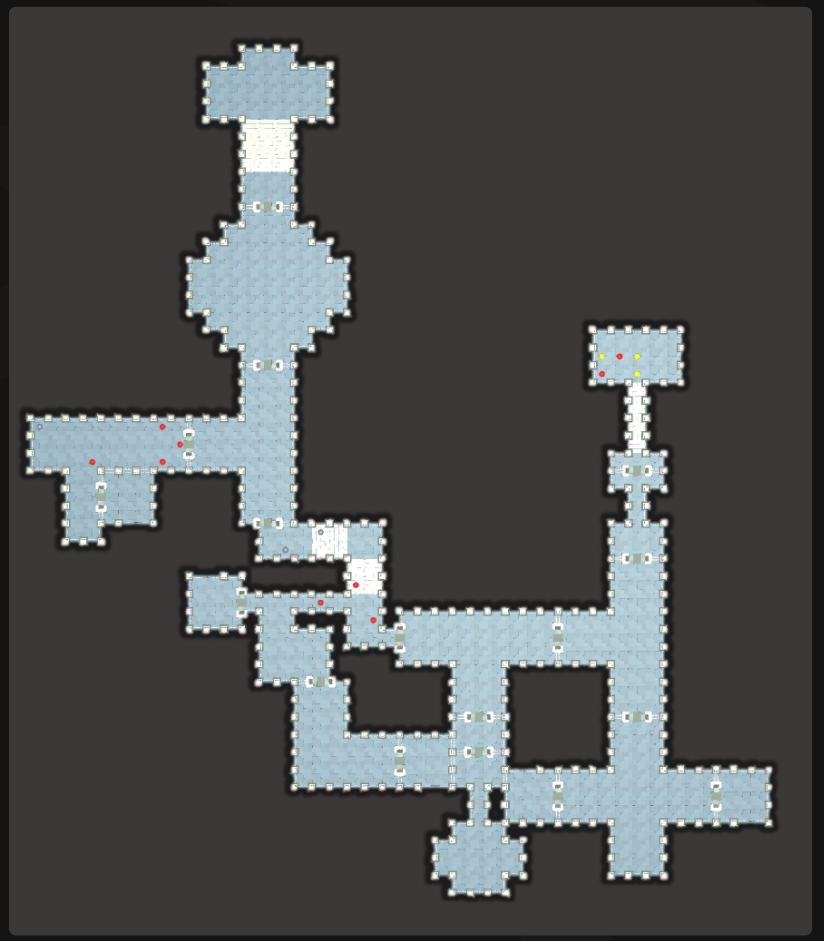
Transform into Level Space (B)

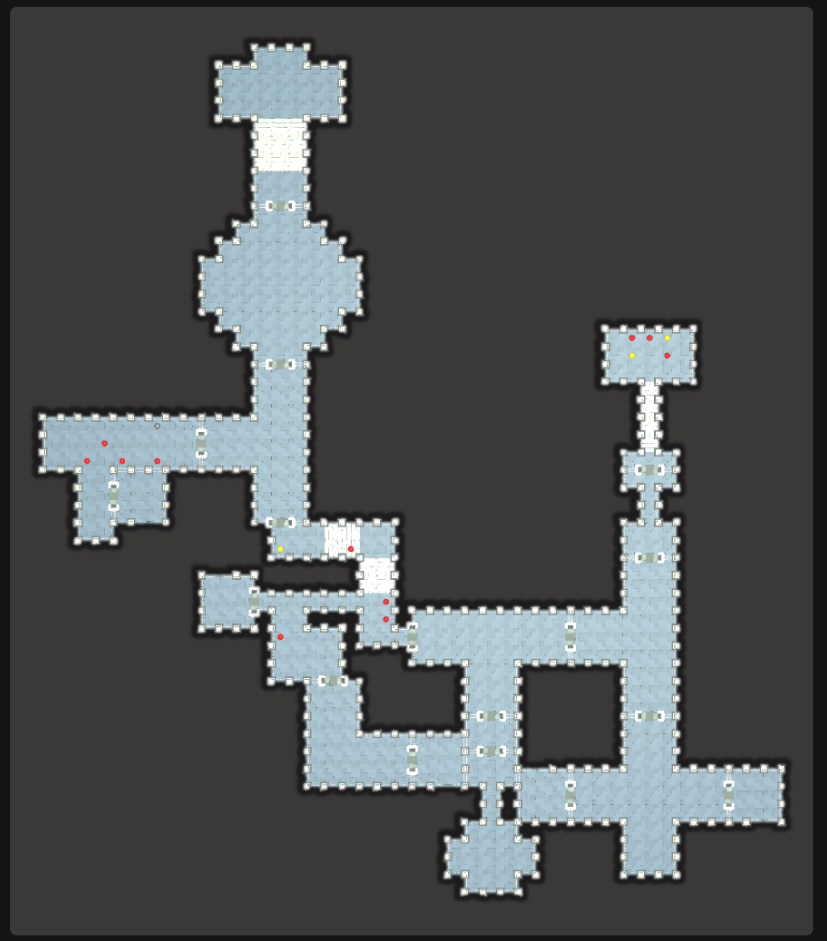




Generate the game objects (B)



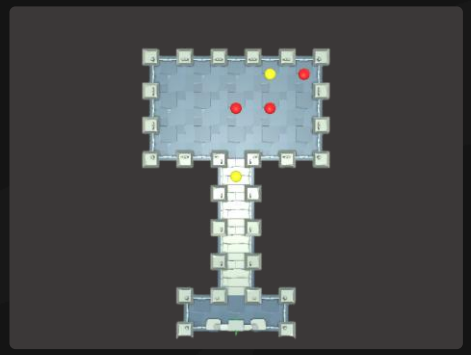




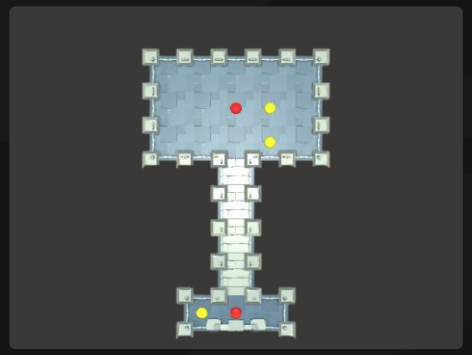
Results of volume

Observe the results each evoluted volume

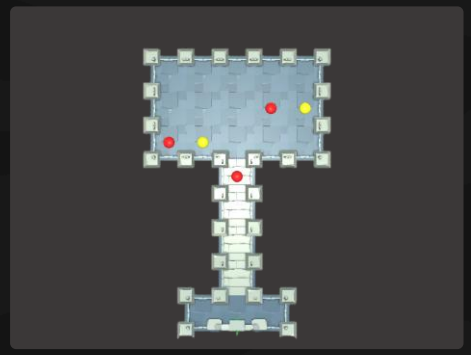
Treasure 1 (100 generations, 50 individuals)



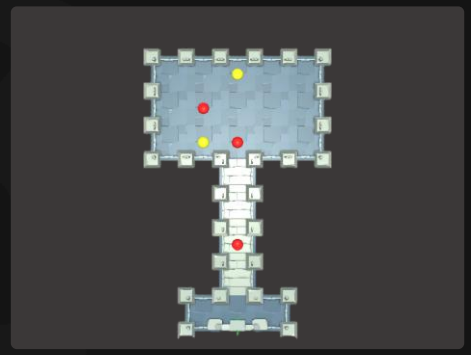
(i) 1589*ms*



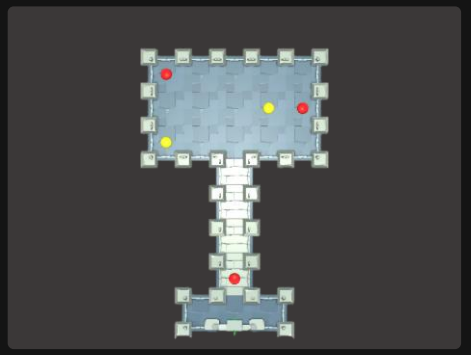
(ii) 1583*ms*



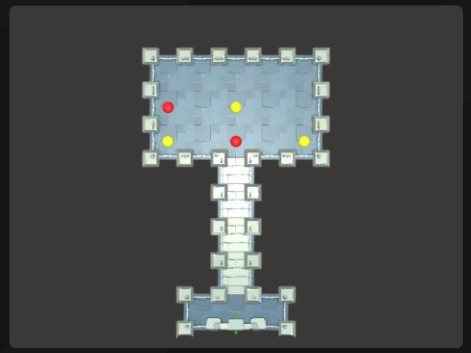
(iii) 1589*ms*



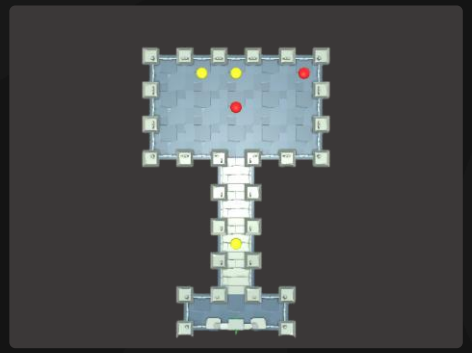
(iv) 1579*ms*



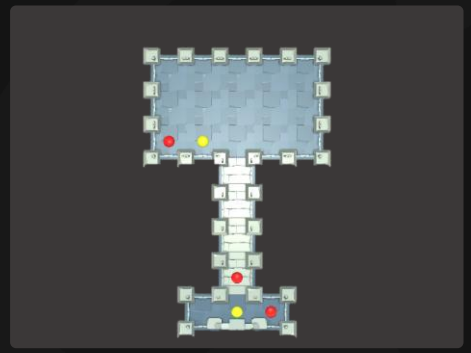
(v) 1587*ms*



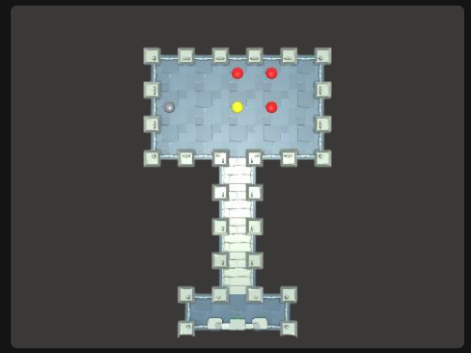
(vi) 1579*ms*



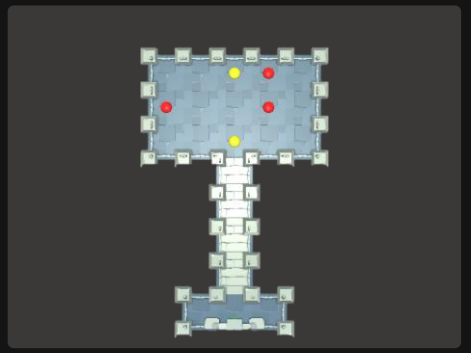
(vii) 1586*ms*



(viii) 1578*ms*

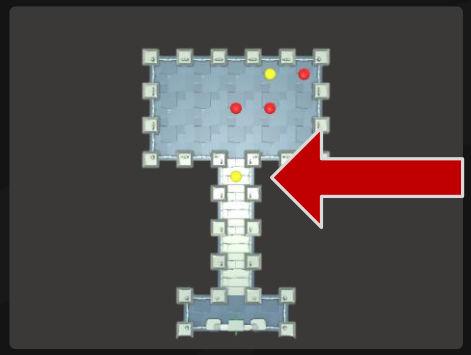


(ix) 1658*ms*

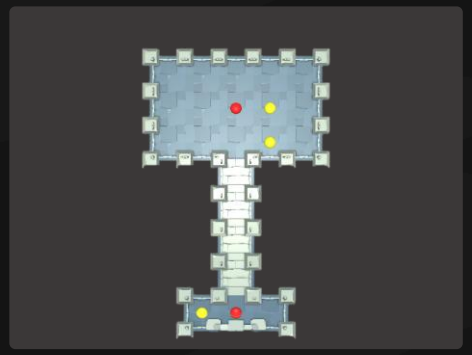


(x) 1612*ms*

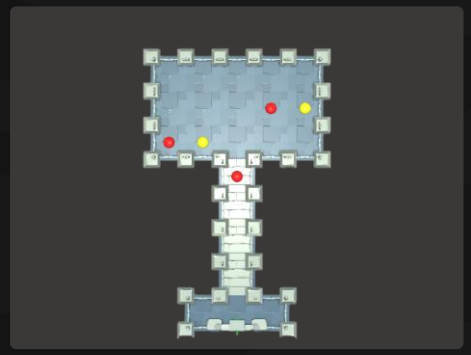
Treasure 1 (100 generations, 50 individuals)



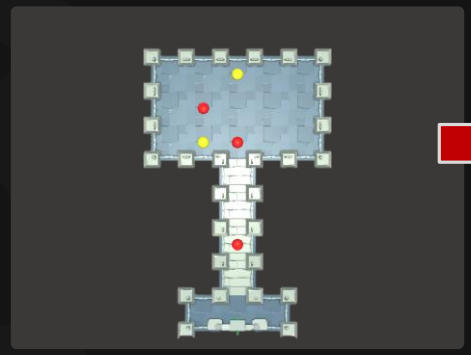
(i) 1589*ms*



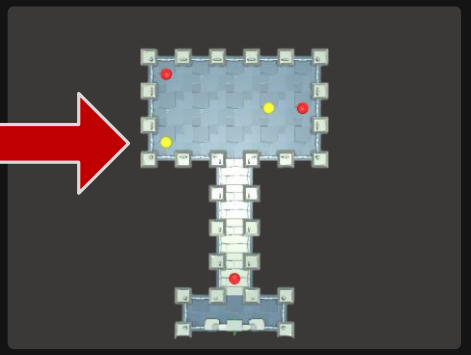
(ii) 1583*ms*



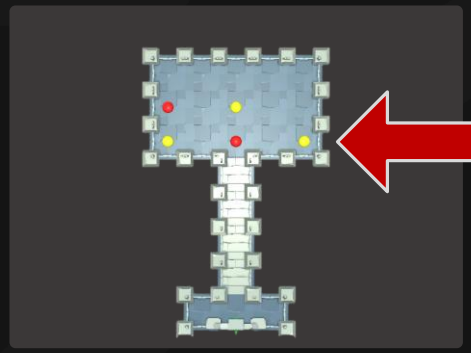
(iii) 1589*ms*



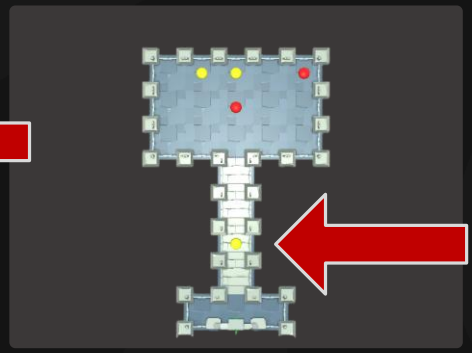
(iv) 1579*ms*



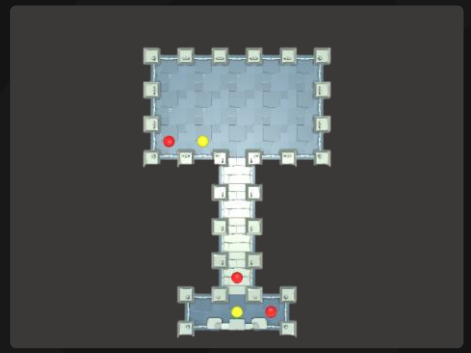
(v) 1587*ms*



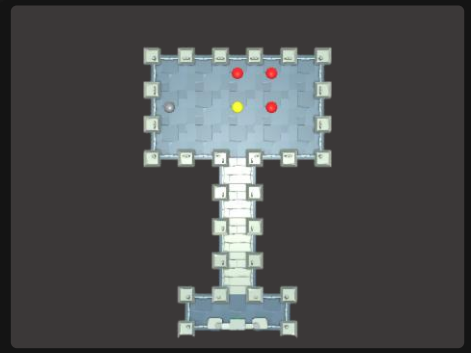
(vi) 1579*ms*



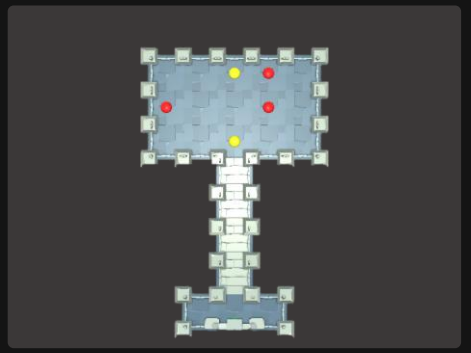
(vii) 1586*ms*



(viii) 1578*ms*

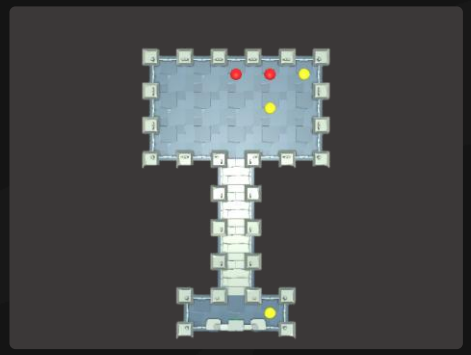


(ix) 1658*ms*

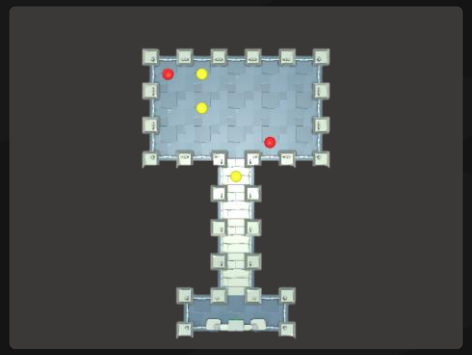


(x) 1612*ms*

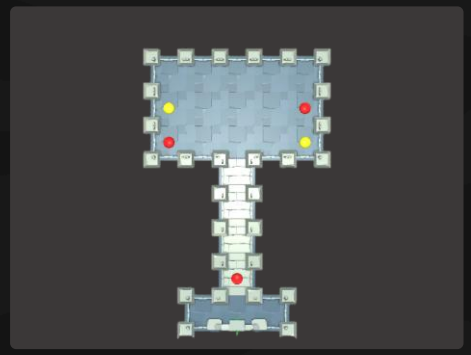
Treasure 2 (100 generations, 100 individuals)



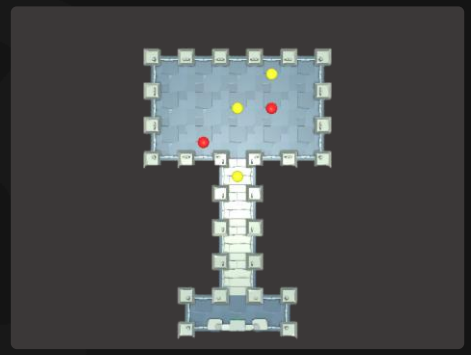
(i) 3223*ms*



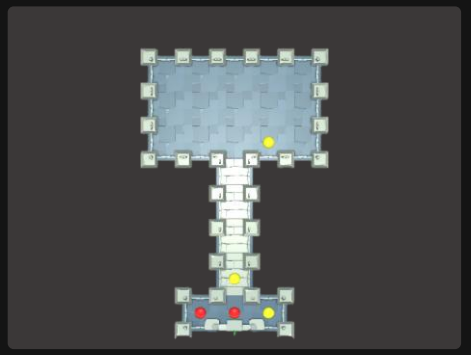
(ii) 3236*ms*



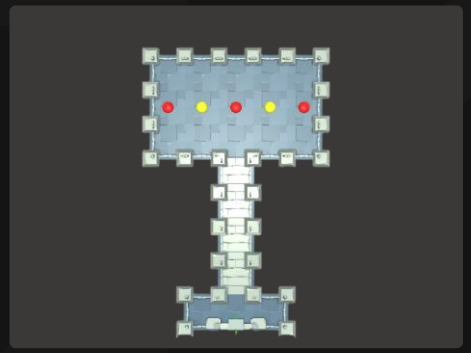
(iii) 3257*ms*



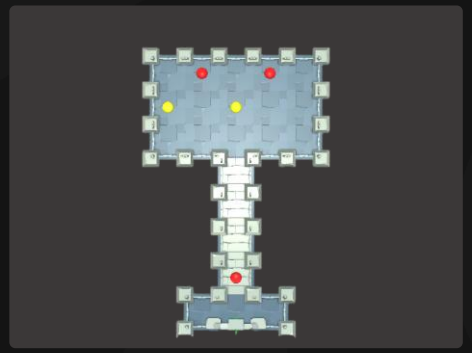
(iv) 3240*ms*



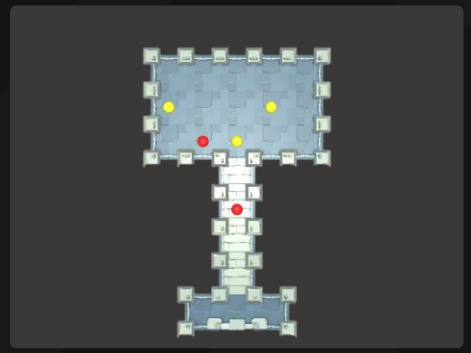
(v) 3244*ms*



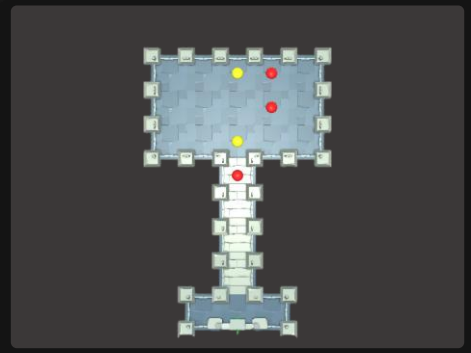
(vi) 3226*ms*



(vii) 3204*ms*



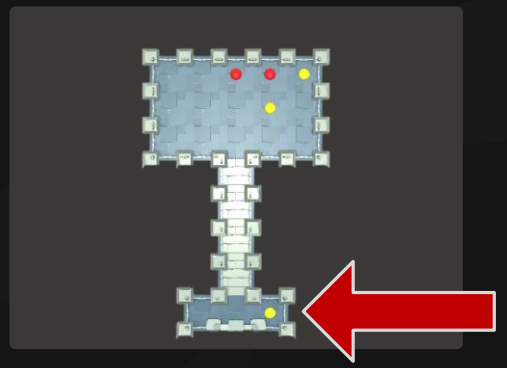
(viii) 3262*ms*



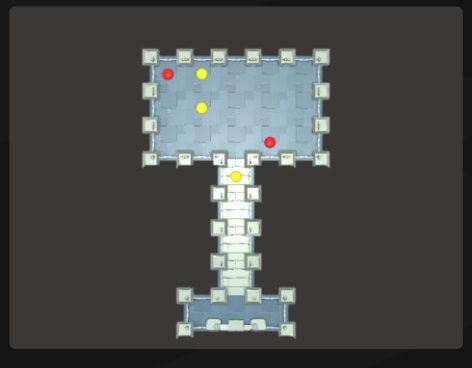
(ix) 3278*ms*

(x) 3240*ms*

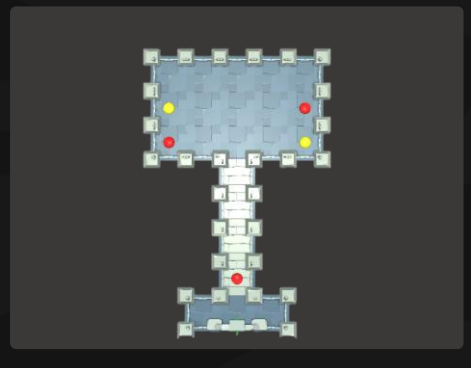
Treasure 2 (100 generations, 100 individuals)



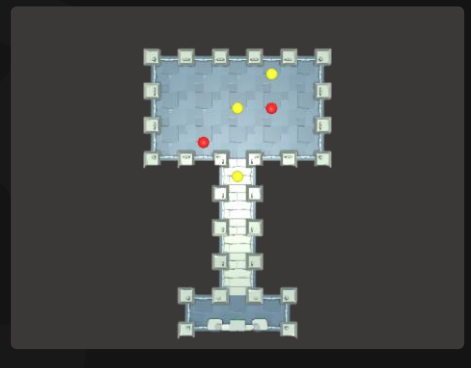
(i) 3223*ms*



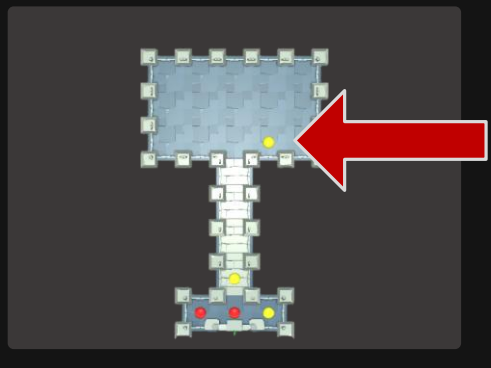
(ii) 3236*ms*



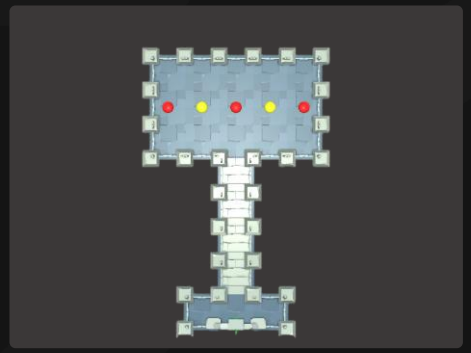
(iii) 3257*ms*



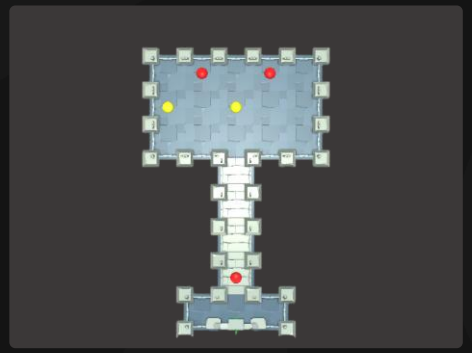
(iv) 3240*ms*



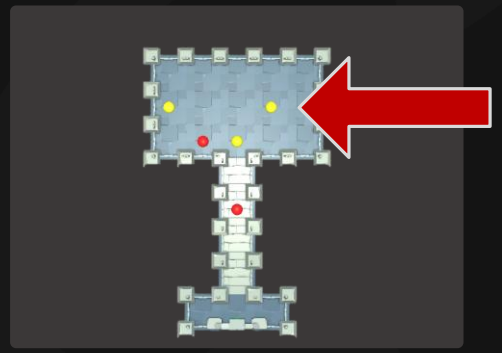
(v) 3244*ms*



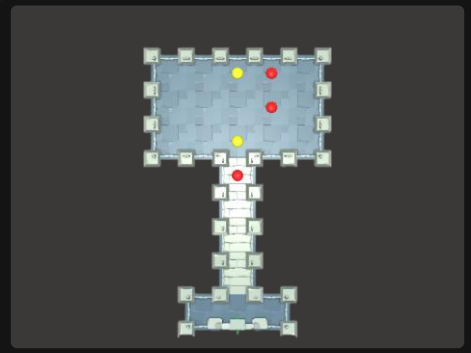
(vi) 3226*ms*



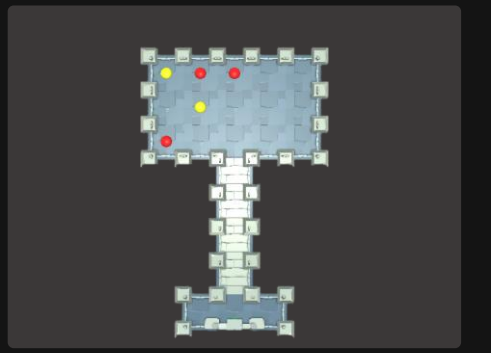
(vii) 3204*ms*



(viii) 3262*ms*

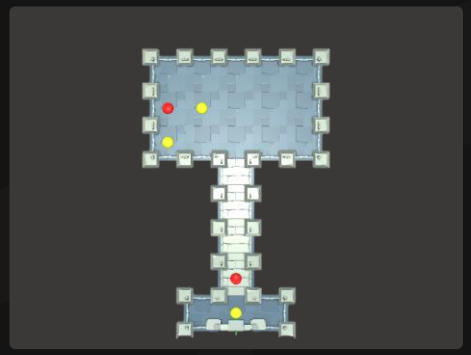


(ix) 3278*ms*

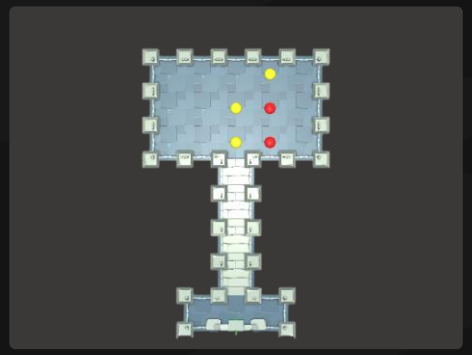


(x) 3240*ms*

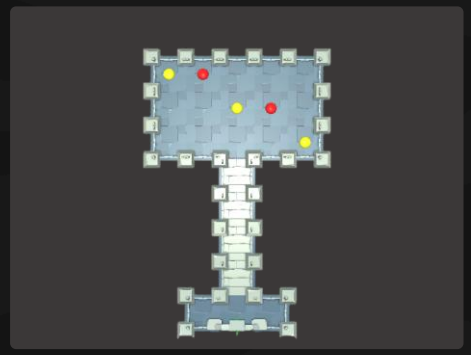
Treasure 3 (100 generations, 200 individuals)



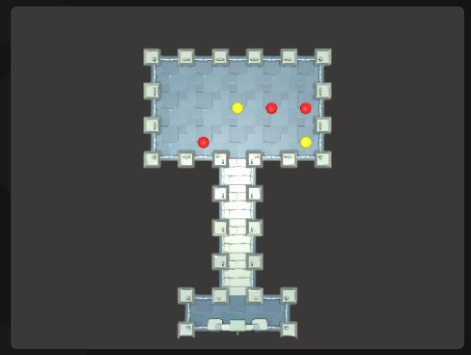
(i) 6949*ms*



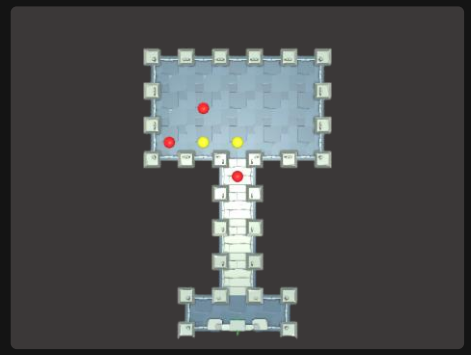
(ii) 6953*ms*



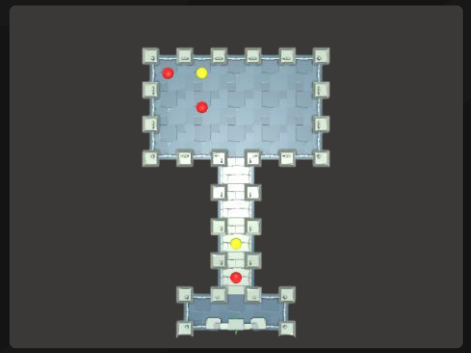
(iii) 7254*ms*



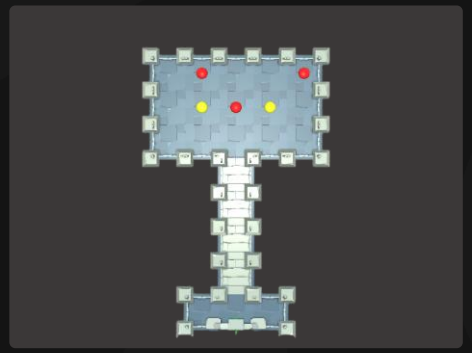
(iv) 6736*ms*



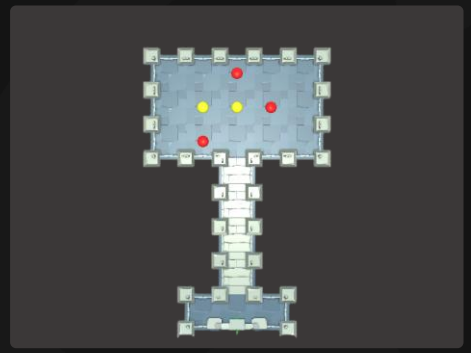
(v) 6729*ms*



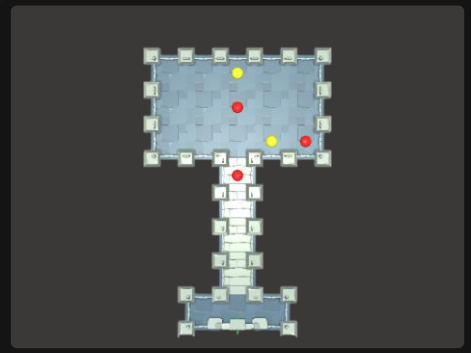
(vi) 6690*ms*



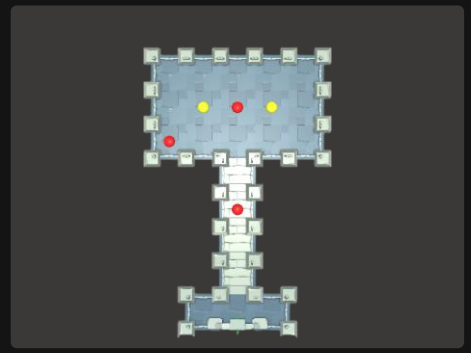
(vii) 6775*ms*



(viii) 6684*ms*



(ix) 6686*ms*



(x) 6654*ms*

Hypothesis

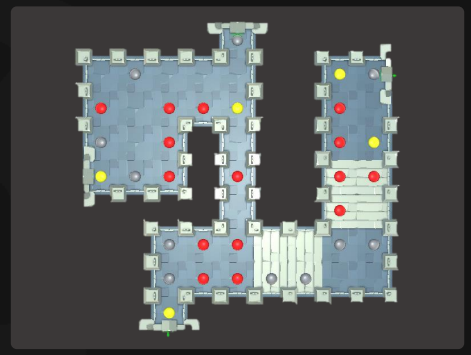
- Size of the population (number of the individual) leads to **local optimum**
  - For “Treasure” volume:
    Some of treasure objects are not close to enemy object.

Narrow 1 (100 generations, 50 individuals)

(i) 3195*ms*

(ii) 3222*ms*

(iii) 3287*ms*



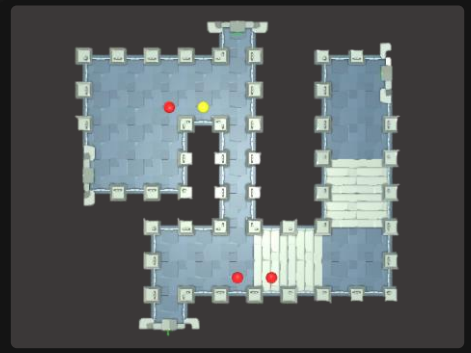
(iv) 3596*ms*

(v) 3218*ms*

(vi) 3242*ms*

(vii) 3179*ms*

(viii) 3165*ms*



(ix) 3256*ms*

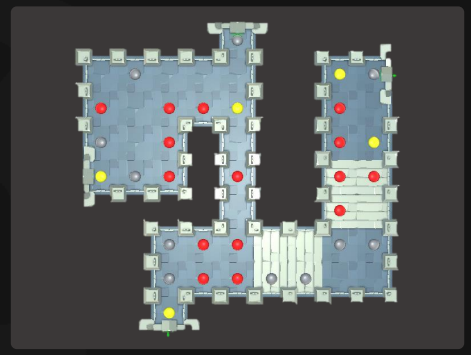
(x) 3194*ms*

Narrow 1 (100 generations, 50 individuals)

(i) 3195*ms*

(ii) 3222*ms*

(iii) 3287*ms*



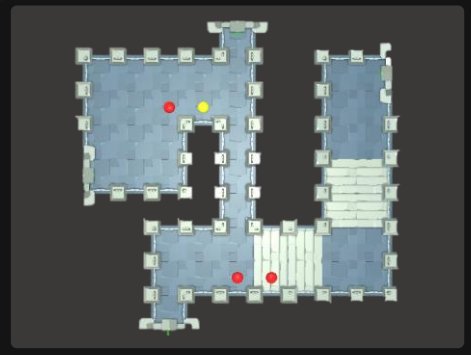
(iv) 3596*ms*

(v) 3218*ms*

(vi) 3242*ms*

(vii) 3179*ms*

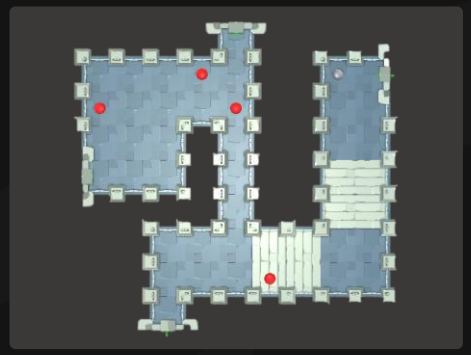
(viii) 3165*ms*



(ix) 3256*ms*

(x) 3194*ms*

Narrow 2 (100 generations, 100 individuals)



(i) 6423*ms*

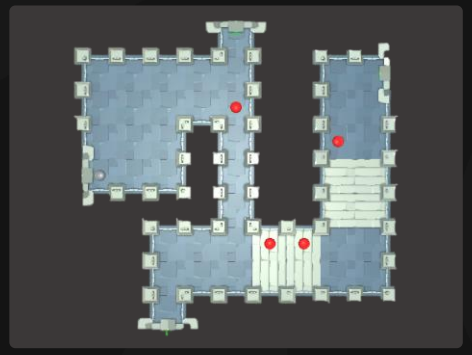
(ii) 6394*ms*

(iii) 6499*ms*

(iv) 6385*ms*

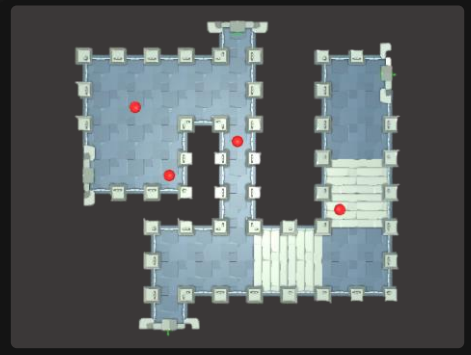
(v) 6363*ms*

(vi) 6441*ms*



(vii) 6410*ms*

(viii) 6366*ms*

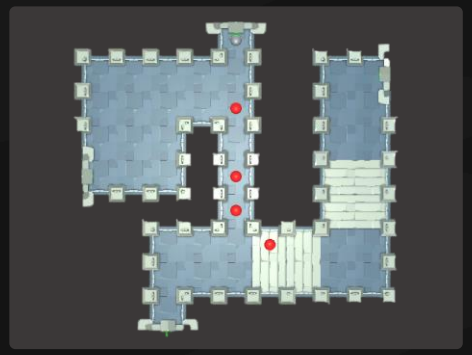


(ix) 6542*ms*

(x) 6364*ms*

Narrow 3 (100 generations, 200 individuals)

(i) 12798*ms*



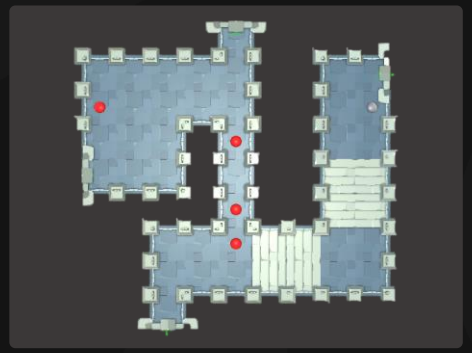
(ii) 12905*ms*

(iii) 12950*ms*

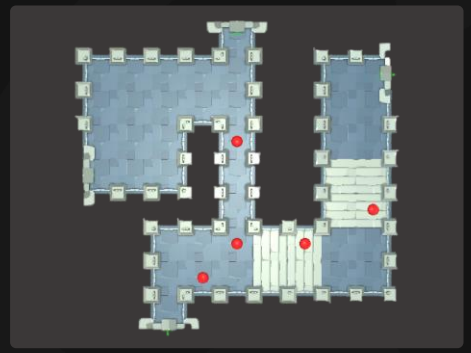
(iv) 12993*ms*

(v) 12874*ms*

(vi) 12877*ms*



(vii) 13085*ms*



(viii) 13161*ms*

(ix) 12829*ms*

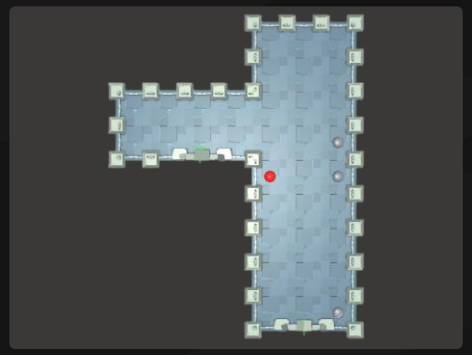
(x) 12998*ms*

Hypothesis

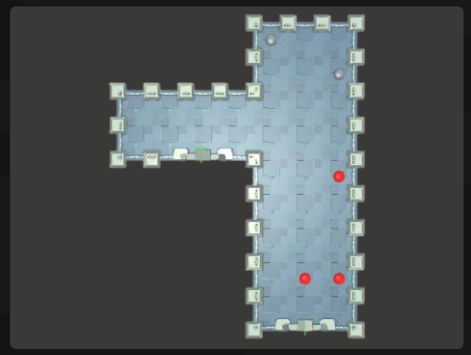
- Size of the population (number of the individual) leads to **local optimum**
  - For “Treasure” volume:
    Some of treasure objects are not close to enemy object.
  - For “Narrow” volume:
    In the last generation, all individual are out of the requirement of *density*.

Trunk 1 (100 generations, 50 individuals)

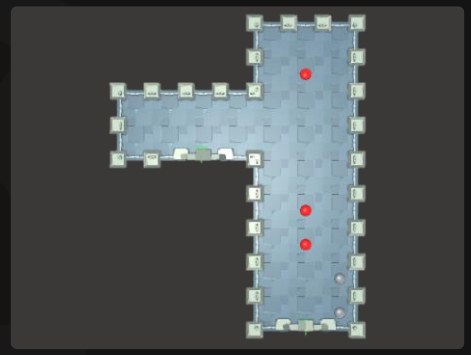
(i) 2662*ms*



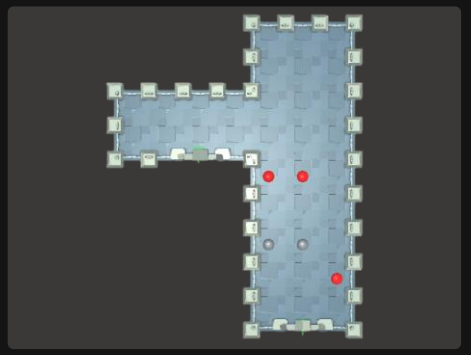
(ii) 2664*ms*



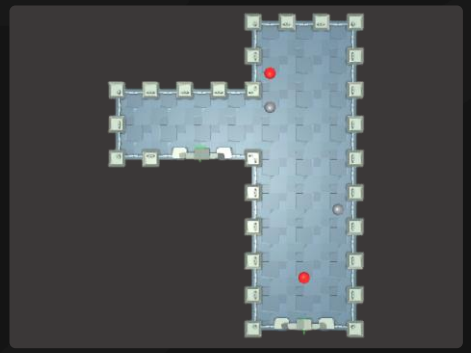
(iii) 2695*ms*



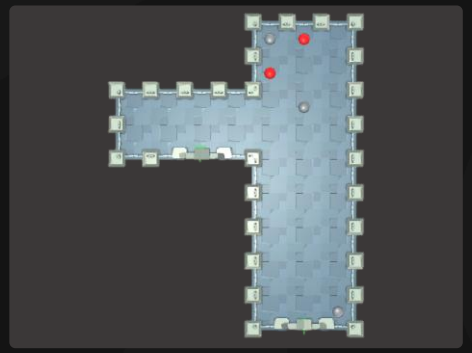
(iv) 2706*ms*



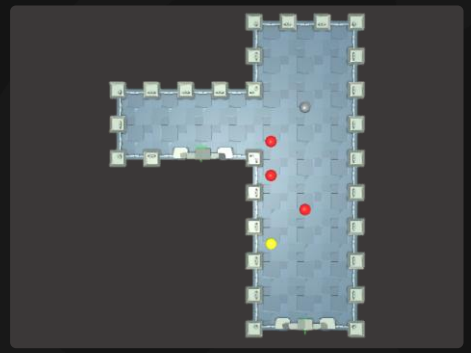
(v) 2731*ms*



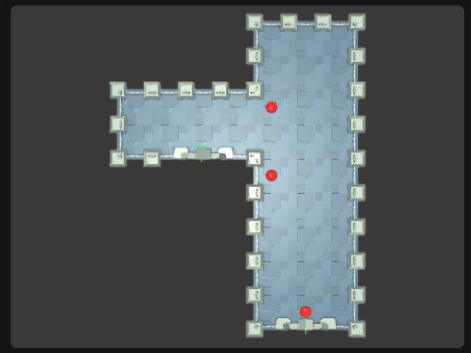
(vi) 2707*ms*



(vii) 2719*ms*



(viii) 2780*ms*

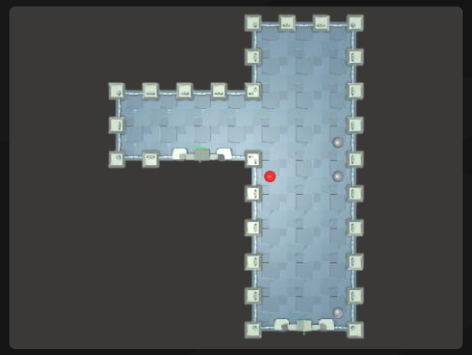


(ix) 2766*ms*

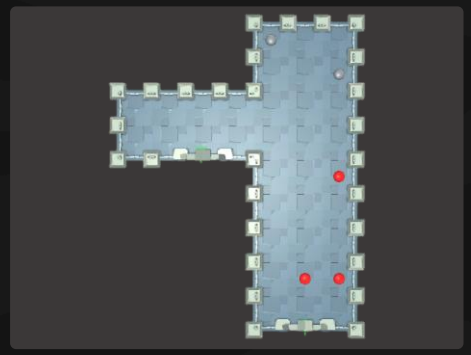
(x) 2649*ms*

Trunk 1 (100 generations, 50 individuals)

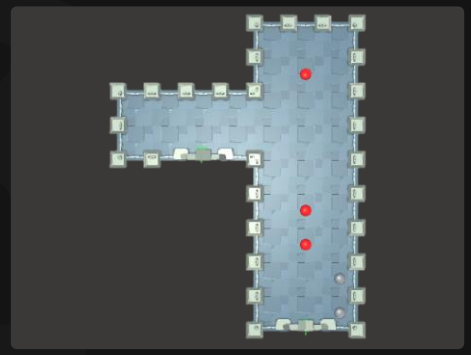
(i) 2662*ms*



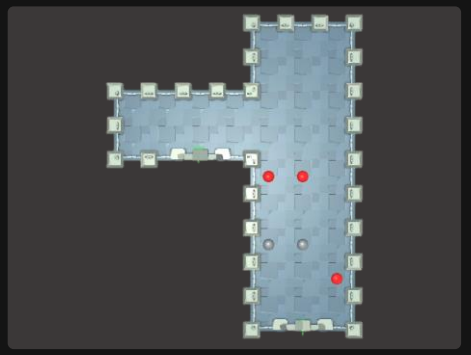
(ii) 2664*ms*



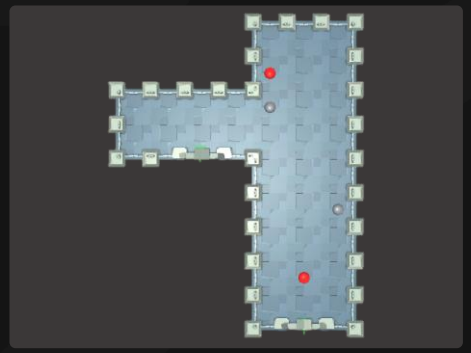
(iii) 2695*ms*



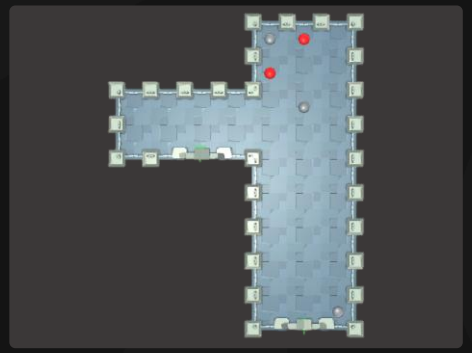
(iv) 2706*ms*



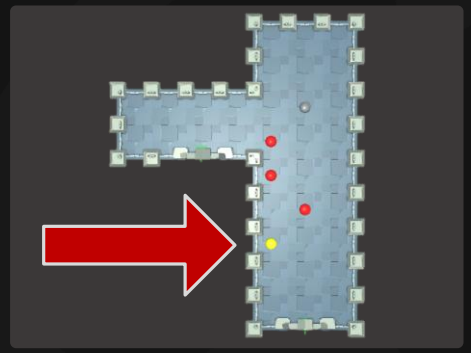
(v) 2731*ms*



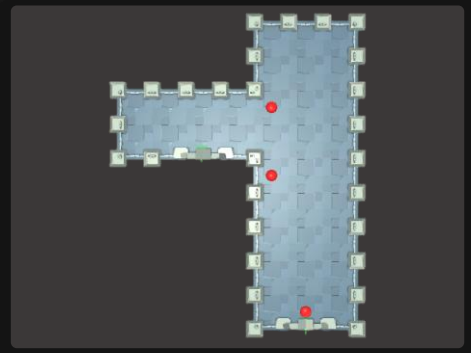
(vi) 2707*ms*



(vii) 2719*ms*



(viii) 2780*ms*



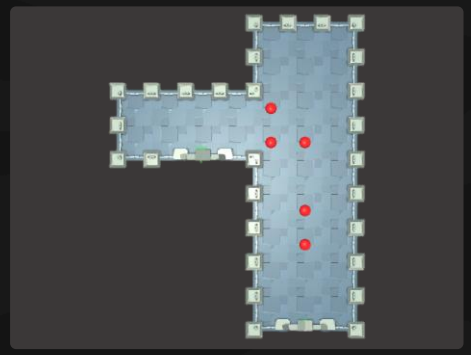
(ix) 2766*ms*

(x) 2649*ms*

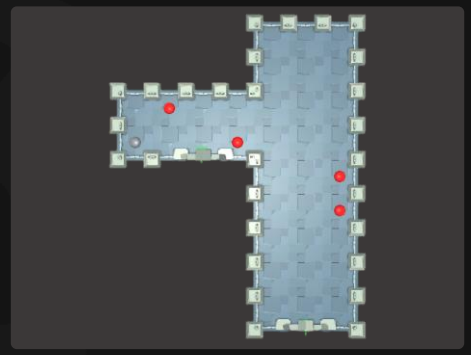
Trunk 2 (100 generations, 100 individuals)

(i) 5413*ms*

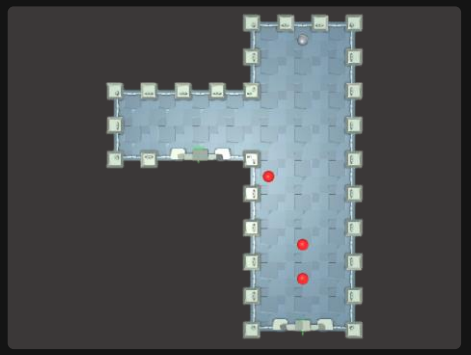
(ii) 5381*ms*



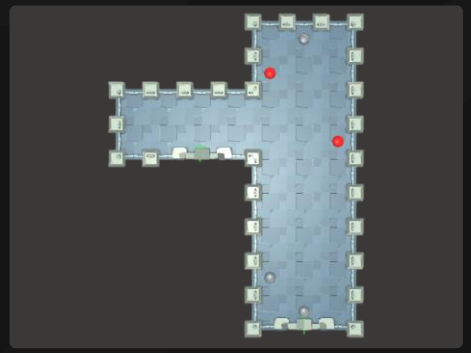
(iii) 5594*ms*



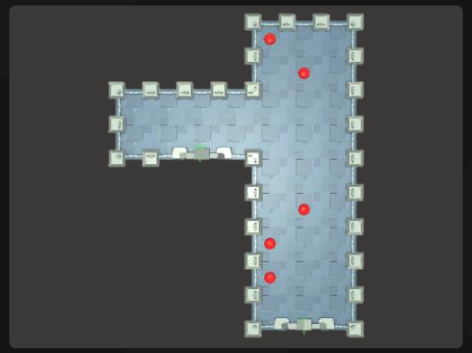
(iv) 5546*ms*



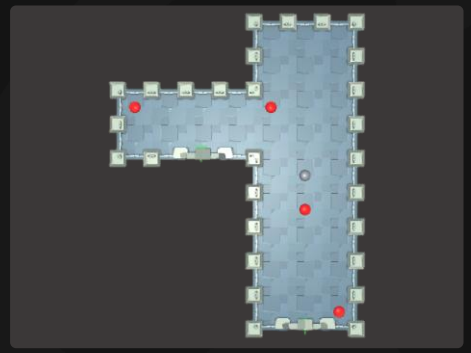
(v) 5460*ms*



(vi) 5365*ms*

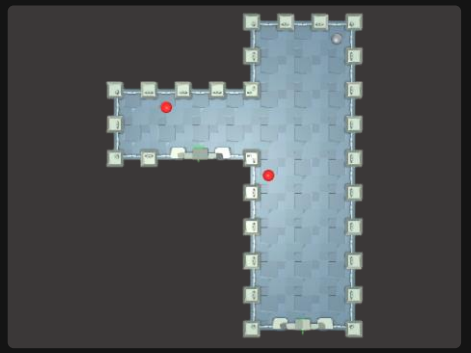


(vii) 5523*ms*



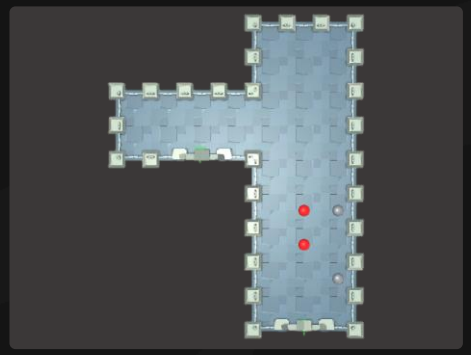
(viii) 5425*ms*

(ix) 5369*ms*

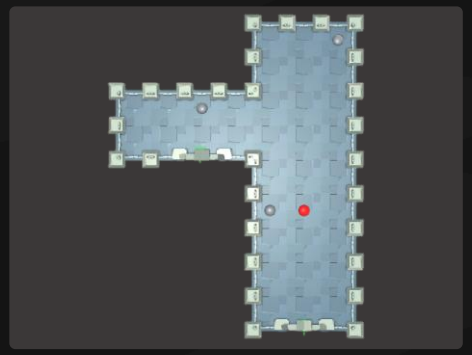


(x) 5310*ms*

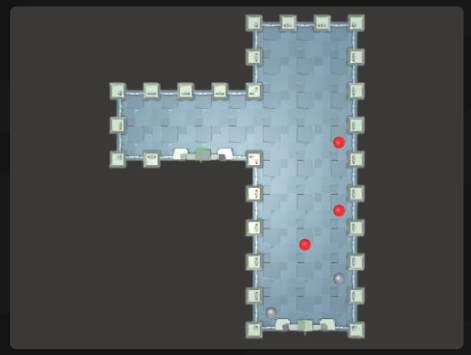
Trunk 3 (100 generations, 200 individuals)



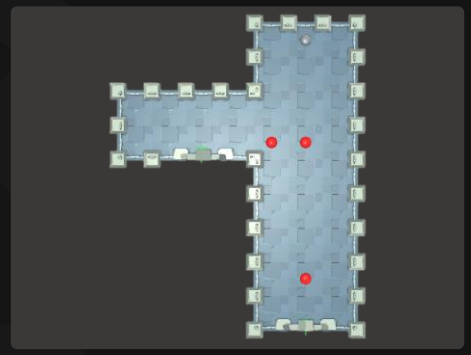
(i) 10972*ms*



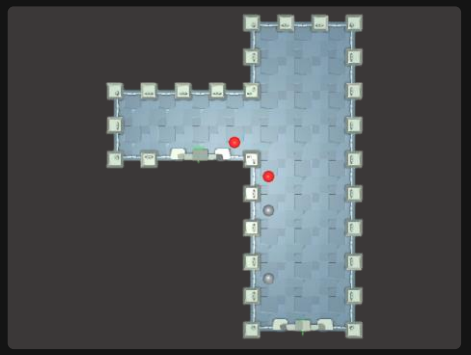
(ii) 10797*ms*



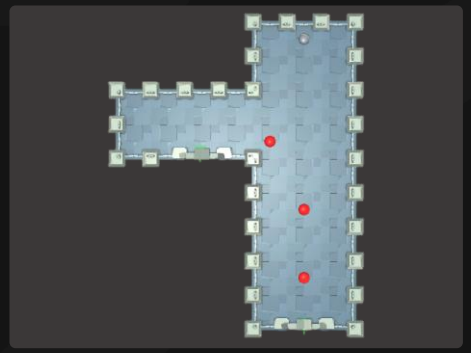
(iii) 11168*ms*



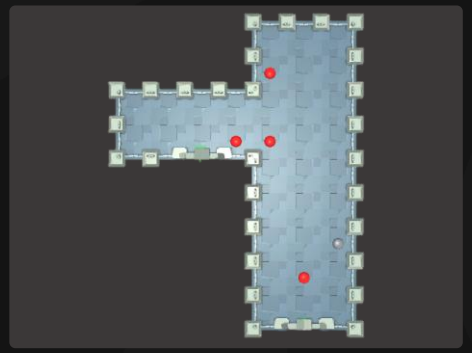
(iv) 10907*ms*



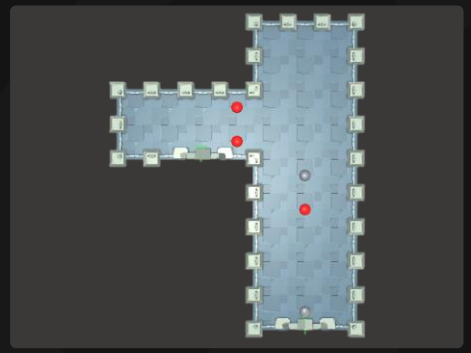
(v) 11035*ms*



(vi) 10981*ms*

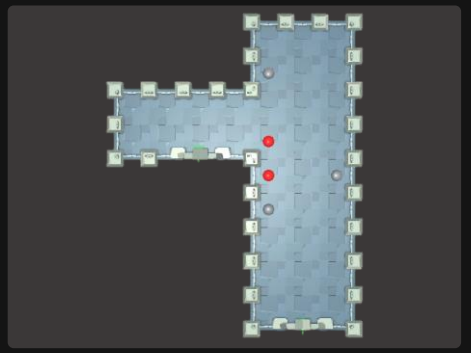


(vii) 11073*ms*



(viii) 10916*ms*

(ix) 10931*ms*



(x) 11052*ms*

Hypothesis

- Size of the population (number of the individual) leads to **local optimum**
  - For “Treasure” volume:
    Some of treasure objects are not close to enemy object.
  - For “Narrow” volume:
    In the last generation, all individual are out of the requirement of *density*.
  - For “Trunk” volume:
    Still appear the treasure object (The weight of *guard* is minus).

Population size	Avg. fitness score	Population standard deviation
50	3.6347	0.42009
100	3.7713	0.29068
200	3.8458	0.12758

DEMO

Gameplay

DEMO

Video here

Conclusions and Contributions

Conclusions

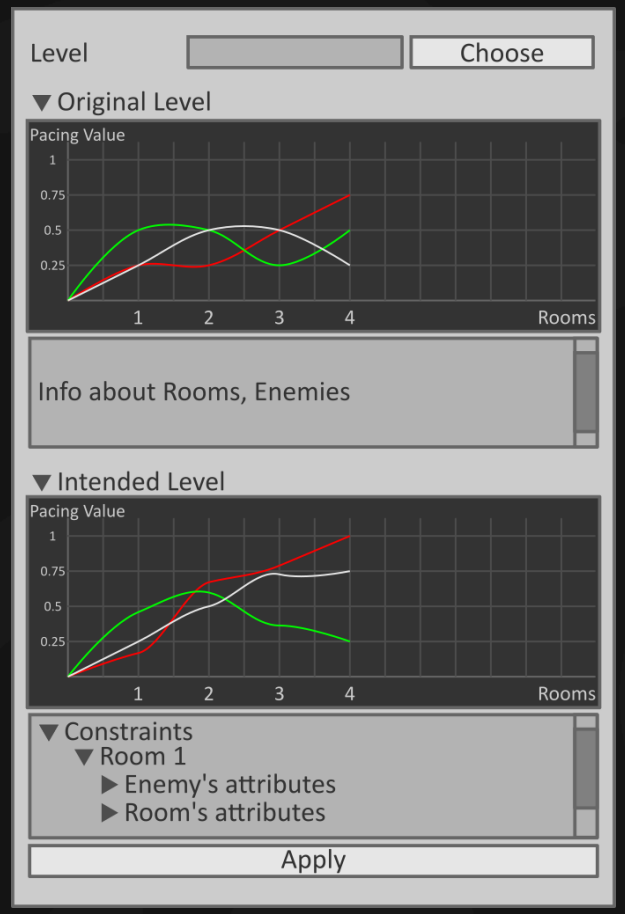
- 藉由 *Dungeon Generator* 高度抽象遊戲概念的遊戲開發輔助工具，主以視覺化其概念邏輯之結構，輔以高度語意化的標籤進行數值微調，大幅降低關卡設計師的設計成本，終以加速關卡的生成產量。
- 由 *Dungeon Generator* 藉由有意義為核心目標進行關卡生成，消彌了市面上絕大多數關卡生成工具的隨機不確定性，便能夠控管玩家的遊玩體驗。

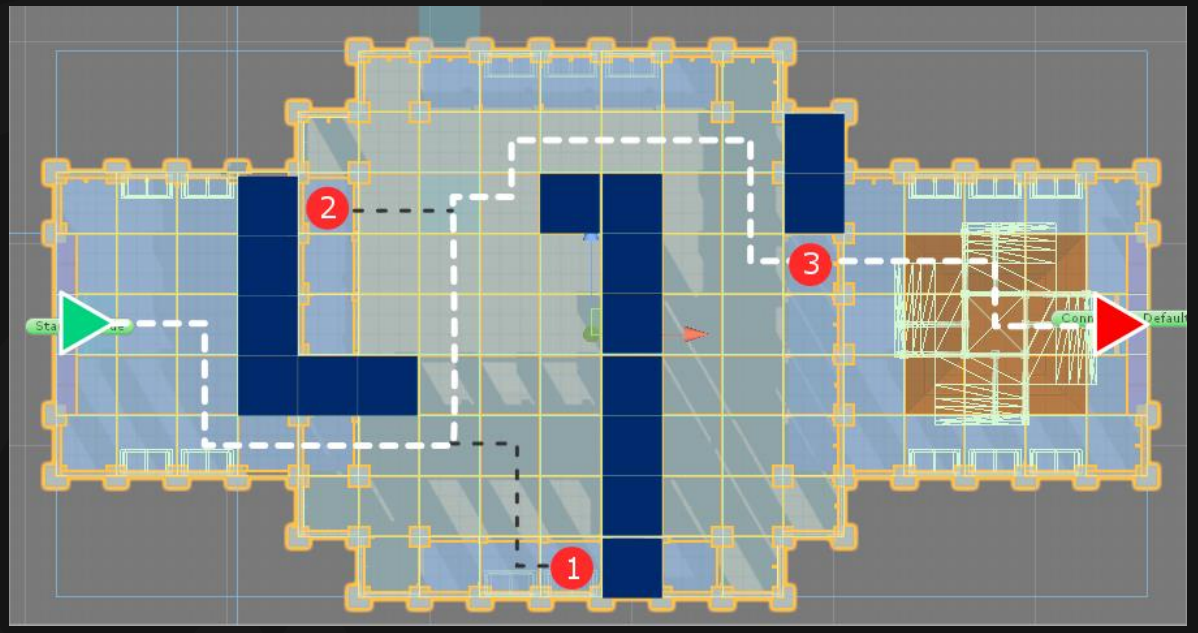
Contributions

- 房間容器依照遊玩特徵指標進行演化的結果，即使透過抽象化（體素化）空間設計其遊玩特徵指標，精確度降低的情形下，其結果仍具有一定品質與遊戲體驗。
- 提出之適應值標準化方法能解決多目標最佳化問題 (Multi-objective optimization)，讓不同的適應值能相互均衡發展與演化。
- 綜合所有方法所彙整的關卡自動生成工具 *Dungeon Generator*，提供了完整流程的關卡生成解決方案。

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).

Thanks for listening

- Special Thanks
  - Advisor: **Prof. Wen-Kai Tai**
  - XAOCX team: Ally, Bagus, Fong, Harrison, Guang-Bo
  - Partners: **Mr. Alex Tsai** and his team