基於強化學習的動態迷宮智能體：以「迷宮之主」實現動態難度調整

蔡易龍

正修科技大學 41311202 蔡易龍

41311202@gcloud.csu.edu.tw

高詮鈞

正修科技大學 41311138 高詮鈞

41211138@gcloud.csu.edu.tw

黎氏嬌縈

正修科技大學 41111148 黎氏嬌縈

41111148@gcloud.csu.edu.tw

Abstract

本專案旨在設計一個基於強化學習（RL）的動態迷宮系統，其核心創新在於將學習智能體（Agent）由「玩家」轉移至「迷宮之主」。系統包含玩家、小怪與「迷宮之主」三個角色。傳統迷宮設計多為靜態或隨機生成，缺乏適應玩家行為的動態調整機制。為解決此問題，我們提出一個「迷宮之主」智能體，其利用強化學習技術，根據玩家的即時狀態（如位置、血量、遊戲進程）動態調整迷宮的出口位置與內部結構（如牆壁）。本研究預計採用深度Q網路（DQN）結合程序化內容生成（PCGRL）與動態難度調整（DDA）技術，訓練智能體學習最佳策略，以期在遊戲過程中產生具持續挑戰性與多樣性的迷宮體驗。

1. Introduction

強化學習（Reinforcement Learning, RL）已被廣泛應用於訓練智能體解決迷宮問題，例如透過 Q-learning 或深度強化學習（DRL）演算法訓練角色尋找最優路徑。然而，這些研究大多將「玩家」或「角色」作為學習的對象，而忽略了遊戲環境本身也可以作為智能體的一部分進行學習與調整。

在傳統遊戲設計中，迷宮地圖往往是靜態生成或基於簡單隨機算法生成，這種設計缺乏對玩家行為的即時反饋與適應性。靜態設計可能導致玩家在熟悉地圖後感到挑戰性不足，或者在初期遭遇過高難度，進而影響整體的遊戲體驗與沉浸感。

為此，本研究提出一個核心問題：如何設計一個「迷宮之主」（Dungeon Master）智能體，使其能夠根據玩家的行為動態生成出口位置、調整迷宮布局，並透過召喚小怪等行為影響遊戲進程，以達到動態難度調整（Dynamic Difficulty Adjustment, DDA）的效果。我們嘗試將強化學習的主體由「玩家」轉移至「迷宮」，讓迷宮本身具備學習與適應能力。

1.1 Existing Solution

本研究建立在兩個主要的相關領域之上：程序化內容生成（PCG）與動態難度調整（DDA）。

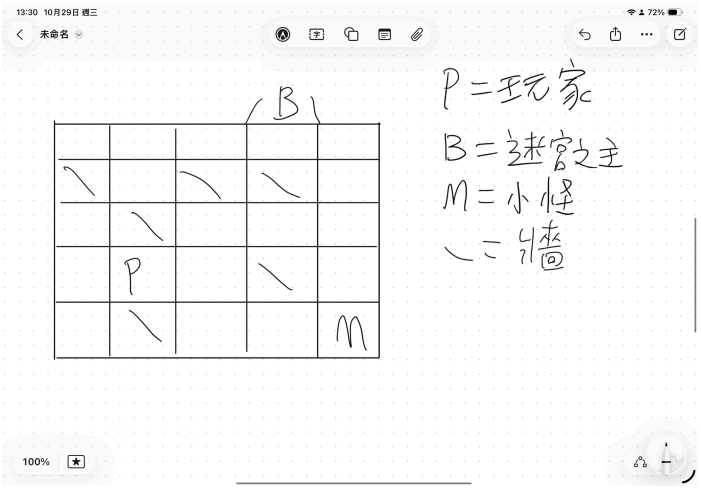
- 程序化內容生成 (PCG):
  PCG 技術允許透過演算法自動生成遊戲關卡，減少人工設計成本並增加多樣性。近年來，PCGRL (Procedural Content Generation via Reinforcement Learning) 提供了一種將地圖生成建模為馬可夫決策過程（MDP）的框架。Agent 透過一系列動作（如放置磚塊）來建構關卡，並根據生成關卡的可玩性獲得獎勵。
- 動態難度調整 (DDA):
  DDA 旨在根據玩家表現即時調整遊戲參數，以維持玩家的「心流」（Flow）體驗。傳統 DDA 多依賴靜態規則腳本（例如：若玩家連續失敗3次，則不召喚小怪或降低迷宮佈局改變頻率）。而基於強化學習的 DDA（RL-based DDA）則嘗試讓 Agent 學習一套動態調整策略，以更精準地適應玩家。

1.2 Expected Solution

本研究的創新點在於將 PCGRL 與 RL-based DDA 結合，使「迷宮之主」不僅能調整現有參數（DDA），更能動態改變環境結構本身（PCGRL），從而提供更深層次的動態遊戲體驗。我們的核心假設是：此結合將能創造出比傳統靜態迷宮更具挑戰性與可玩性的遊戲體驗。

2. Method

我們的目標是訓練一個「迷宮之主」智能體（Agent）。本節將詳細描述系統架構、馬可夫決策過程（MDP）的定義，以及我們選用的演算法。



2.1 Architecture

我們設計一個閉環（Closed-loop）學習系統，其運作流程如下：

1. 玩家行動：玩家在$N \times N$的迷宮環境中移動，其行為（移動、使用道具、被攻擊）會改變環境狀態。
2. 狀態觀測：玩家每行動 k 步（例如 $k = 5$），「迷宮之主」Agent 觀測一次當前的全局狀態 s_t。
3. **Agent** 決策：Agent 的神經網路模型（DQN）根據狀態 s_t 輸出一系列動作的 Q-value，並依 $\epsilon - greedy$策略選擇一個動作 a_t（例如：在 (x,y) 放置牆壁）。
4. 環境更新：環境執行動作 at，更新為新狀態 s_{t+1}（例如迷宮結構改變）。
5. 獎勵計算：系統根據 s_{t+1} 所體現的遊戲挑戰度，計算一個即時獎勵 r_t 並回傳給 Agent。
6. 經驗儲存與學習：將 (s_t, a_t, r_t, s_{t+1}) 存入經驗回放池（Replay Buffer）中，用於 Agent 的離線訓練。

2.2 馬可夫決策過程 (MDP) 定義

我們從「迷宮之主」Agent 的視角來定義 MDP：

- 狀態 (State, S):
  我們採用完全資訊的狀態表示法。狀態 s_t 是一個 $N \times N \times C$ 的張量(Tensor)，其中 $N \times N$ 是迷宮尺寸，先暫定為 15×15:
  - C_1: 牆壁位置 (0=空, 1=牆)
  - C_2: 玩家位置 (0=無, 1=玩家)
  - C_3: 小怪位置 (0=無, 1=小怪)
  - C_4: 出口位置 (0=無, 1=出口)
  - C_5: 玩家血量 (歸一化值 0.0 ~ 1.0)
- 動作 (Action, A):
  Agent 的動作空間是離散的。為了簡化問題，我們定義的動作 a_t 包含:
  - **No-op**:不執行任何動作。
  - 改變出口:將出口移動到指定 M 個候選位置一。
  - 改變牆壁:在玩家鄰近 $k \times k$ 範圍內的 (x,y) 座標生成一面牆。
  - 消除牆壁:在玩家鄰近 $k \times k$ 範圍內的 (x,y) 座標消除一面牆。
  - 召喚小怪:在玩家鄰近 $k \times k$ 範圍內的 (x,y) 座標召喚一隻小怪。
- 獎勵 (Reward, R):
  獎勵函數 R_t 的設計旨在平衡遊戲的挑戰性:
  - 維持挑戰 **(正獎勵)**:
    - 玩家被小怪攻擊 (非致命): $R \mathrel{+}= 5$
    - 玩家通關時間在「理想區間」$[T_{min}, T_{max}]$ 內: $R_{final} = +100$
  - 懲罰無聊/過難 **(負獎勵)**:
    - 玩家過快通關($T < T_{min}$): $R_{final} = -100$
    - 玩家受困過久($T > T_{max}$): $R_{final} = -100$
    - 玩家在同區域停留過久(e.g., > 30秒): $R = -1$
    - 玩家血量過低(e.g., < 20%): $R = -5$ (避免難度過高)

2.3 Algorithm

我們分析了幾種潛在的 DDA 實現方法(如表 1 所示)。考量到我們高維度的狀態空間($N \times N$ 地圖)與對學習能力的需求，我們最終選擇使用深度Q網路(DQN)。

表 1:不同 **DDA** 實現方法的比較

方法選項	優點	缺點 / 為何不選
靜態腳本	實現簡單(e.g., IF-THEN)	模式固定、易被識破、無學習能力
DQN (選用)	可行:可用 CNN 處理高維狀態	需大量模擬訓練、調參複雜
策略梯度 (PG)	可行:可學習隨機策略	訓練變異數 (variance) 高、收斂稍慢

我們選用 DQN作為「迷宮之主」Agent 的核心演算法。我們將使用一個卷積神經網路(CNN)來自動提取 $N \times N$ 狀態地圖的特徵，網路的輸出層為一個全連接層，其節點數對應動作空間 N 的維度，輸出每個動作的 Q-value。

3 Experiment

為了驗證我們方法的有效性，我們設計了以下實驗流程。

3.1 Environment

我們將比較兩個版本的迷宮：(A) 傳統靜態迷宮（固定布局），以及 (B) 由我們訓練好的 DQN「迷宮之主」動態調整的迷宮。

3.2 Training Data

強化學習不需要靜態資料集，而是依賴於與模擬環境的互動。

- 玩家模擬 **(Bot)**：在訓練階段，為了產生大量的訓練資料，「玩家」角色將由一個模擬智能體（Bot）扮演。
- **Bot** 演算法：我們使用 A* (A-star) 尋路演算法來模擬玩家。此 Bot 的目標是「不惜一切代價盡快找到出口」，它將作為「迷宮之主」Agent 的訓練對手。

3.3 Metrics

評估指標將包括：

1. 平均通關時間：我們預期 (B) 的通關時間會高於 (A)，但標準差會更小，顯示難度被穩定控制。
2. 玩家存活/死亡率：(B) 應能將存活率控制在一個目標區間（e.g., 60%-80%），而 (A) 可能會極高或極低。
3. 玩家受困次數：(B) 應顯著低於一個「過難」的靜態迷宮。
4. 人類玩家滿意度：(若時間允許) 我們將招募受試者分別遊玩 (A) 和 (B)，並填寫關於挑戰性、趣味性與挫折感的李克特量表問卷。

4 Appendix

為了強化本研究之實作可行性與參考性，以下整理與本專案主題（強化學習、迷宮生成、動態難度調整）高度相關的開源實例與框架：

4.1相關開源框架

OpenAI Gymnasium：建立強化學習環境的標準框架，支援自定義迷宮與觀測空間。
🔗 https://gymnasium.farama.org

• Stable-Baselines3：整合多種 RL 演算法（如 DQN、PPO、A2C），可用於訓練「迷宮之主」智能體。

🔗https://github.com/DLR-RM/stable-baselines3

• PCGRL (Procedural Content Generation via RL)：提供以強化學習生成遊戲關卡的範例，與本研究概念一致。
🔗https://github.com/smearle/control-pcgrl

• MazeLab：可快速建立與可視化格子型迷宮環境，支援與 Gym 整合。

🔗https://github.com/zuoxingdong/mazelab

• TensorBoard / Weights & Biases：可用於記錄訓練過程、Reward 曲線與學習曲線。
🔗https://www.tensorflow.org/tensorboard

4.2相關研究與實驗實例

• PCGRL 迷宮生成案例：Smearle 等人以 RL 為基礎生成遊戲關卡的研究，展示智能體能根據玩家行為調整關卡結構。

🔗https://ojs.aaai.org/index.php/AIIDE/article/view/7416/7341

• RL-based Dynamic Difficulty Adjustment：Andrade 等人（2006）提出的動態遊戲平衡實驗，作為傳統規則式 DDA 的經典對照研究。

🔗https://dl.acm.org/doi/10.1145/1178823.1178871

• Unity ML-Agents 迷宮訓練範例：Unity 官方提供之 3D 迷宮訓練環境，展示 DQN 智能體如何學習探索與找出口。

🔗https://github.com/Unity-Technologies/ml-agents

• Memory Maze (長期記憶迷宮)：Jurgis Petraitis 所設計的 Memory Maze，用於測試 RL 智能體在複雜迷宮中的記憶與策略。

🔗https://github.com/jurgisp/memory-maze

以上框架與實例可作為本專案在實作階段的參考依據。特別是 PCGRL 與 Stable-Baselines3，可作為訓練「迷宮之主」智能體的主要工具，而 MazeLab 與 Unity ML-Agents 則可提供視覺化環境與模擬平台，以利後續系統驗證與展示。

5 Reference

[1] Summerville, A., Snodgrass, S., Guzdial, M., et al. (2018). "Procedural content generation via reinforcement learning (PCGRL)." *IEEE Conference on Games (CoG)*.

[2] Andrade, G., Ramalho, G., Santana, H., and Corruble, V. (2006). "Dynamic game balancing: An evaluation of user satisfaction." *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*.

[3] Yannakakis, G. N., & Togelius, J. (2018). *Artificial Intelligence and Games*. Springer.

[4] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). "Human-level control through deep reinforcement learning." *Nature*, 518(7540), 529–533.

[5] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). "Curiosity-driven exploration by self-supervised prediction." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.