1) Project Description

1-1) Program Flow Chart

判斷my color

↓

將現在局勢的board去跑4層的[minimax algorithm](https://en.wikipedia.org/wiki/Minimax)

↓

得到可下有效棋最終不同的局面

↓

利用board evaluator去加權判斷局勢

↓

回推決定要下什麼棋

1-2) Detailed Description

Point\* get\_valid\_orbs(Board\* board, char color, int\* count)

→ 將此局面下的有效棋紀錄於Point\* valid\_orbs

void create\_new\_board(Board\* newboard, Point p, char color, Board\* board)

→ 將新的棋子加入後生成newboard

int minimax(Board\* board, int depth, int alpha, int beta, bool isMaximizingPlayer, Point\* ret\_p)

→ 基本上參照wiki的pseudocode

一張含有 文字 的圖片

自動產生的描述

只有為了能回傳point所以Max()跟min()的部分做了調整。

2) Screen Shots

2-1) Partial Implemented Code

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

2-2) GitHub Control History

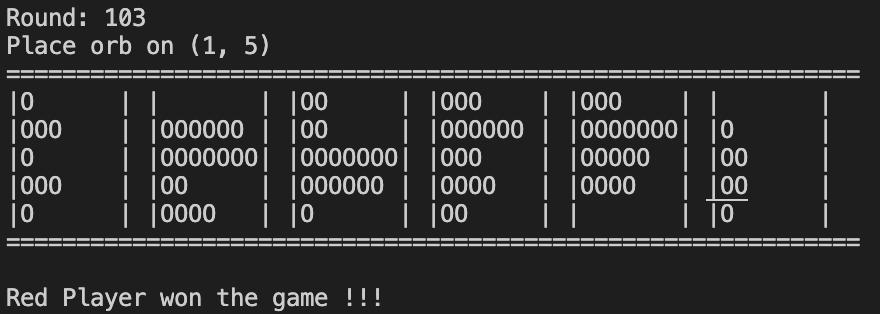
一張含有 文字 的圖片

自動產生的描述

2-3) Compare with TA’s AI Code (*random Move*) for 7 results. (7 pictures)

一張含有 文字, 計分板 的圖片

自動產生的描述一張含有 文字, 計分板 的圖片

自動產生的描述

一張含有 文字, 計分板 的圖片

自動產生的描述一張含有 文字, 計分板 的圖片

自動產生的描述

一張含有 文字, 計分板, 裝置, 儀錶 的圖片

自動產生的描述一張含有 文字, 計分板 的圖片

自動產生的描述

2-4) Describe the reason why you win TA’s AI Code or why you can’t win TA’s AI Code.

基本做法就是利用minimax的game tree加上board evaluator。

而board evaluator內，是簡單利用是否到的達到explode的差異數。

若是對手快到達數量，則減少weight，反之，則加重weight。

因此可以明顯知道，此作法能夠包圍隨機的*random Move*。

以下提供利用script測出的勝率：

一張含有 桌 的圖片

自動產生的描述