사목놀이 과제 보고서



2018. 12. 13

201511182 김 성 헌

201511227 임 준 혁

201310932 정 찬 영

개요 및 함수소개

저희는 이번 프로젝트를 예외처리, 상황평가 그리고 알고리즘의 세가지의 부분으로 나누어 진행하였습니다. 총 4가지의 함수가 정의되었습니다.

1. 예외처리

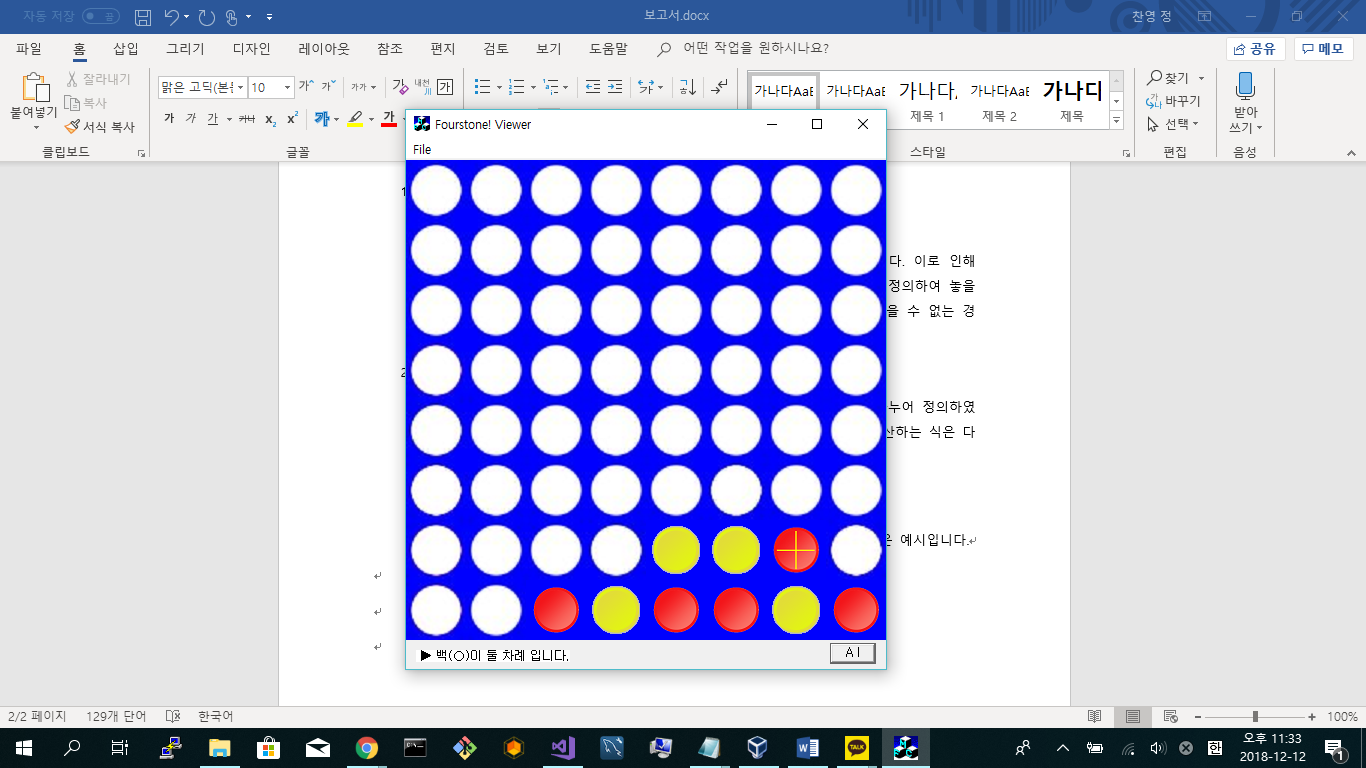
이기는 경우는 exception을 반환하거나 육안으로 확인합니다.

사목놀이에서는 놓고자 하는 자리 아래에는 빈자리가 있어서는 안됩니다. 이로 인해 놓고자 하는 수가 제한되므로 저희는 “possible\_n” 이라는 이름의 함수를 정의하여 놓을 수 있는 수의 좌표를 array의 index와 value를 통하여 표현하였습니다. 놓을 수 없는 경우에는 해당 index의 value 값이 -1 로 고정됩니다. 반환하는 형식은 다음과 같습니다.

1. 상황평가

“evaluate” 라는 함수를 정의하였습니다. 적의 점수와 본인의 점수를 나누어 정의하였고 수들의 연결이라는 특징에 입각하여 함수를 정의하였습니다. 점수를 계산하는 식은 다음과 같습니다.

(#) 은 그 자리에 수를 놓았을 때 연결되는 수의 개수를 의미합니다. 만약 #만큼 연결되어 있는 수가 3개가 있다면 (#)의 값은 3 이 됩니다. 다음은 예시입니다.



만약 위와 같은 수를 놓았을 경우, 우 최하단으로부터 좌로 3, 위로 3인 자리의 점수는

입니다. 평가함수 설명에서 조금 더 자세히 다루겠습니다.

1. 알고리즘

minmax 알고리즘을 사용하여 진행하였습니다. 논문에서 제시되었던 7x7의 게임도 의 경우의 수를 가지고 있었는데 하물며 저희의 게임은 8x8이므로 더욱 대단한 경우의 수를 가지고 있으며, 따라서, 경우의 수를 줄이기 위하여 불능(illegal) 등의 조건을 활용하여 수식적이고 방법론적인 pruning을 시도하였지만 성공하지 못하였습니다. 그로 인해 저희는 알고리즘의 신뢰성에 집중하여 프로젝트를 진행하였습니다.

알고리즘은 “AiChooseColumn” 그리고 “Minmax”라는 이름으로 정의되었습니다. “Minmax”는 min과 max layer 가 구현된 함수이고 “AiChooseColumn” 은 트리를 따라 알고리즘을 진행시켜주는 함수입니다.

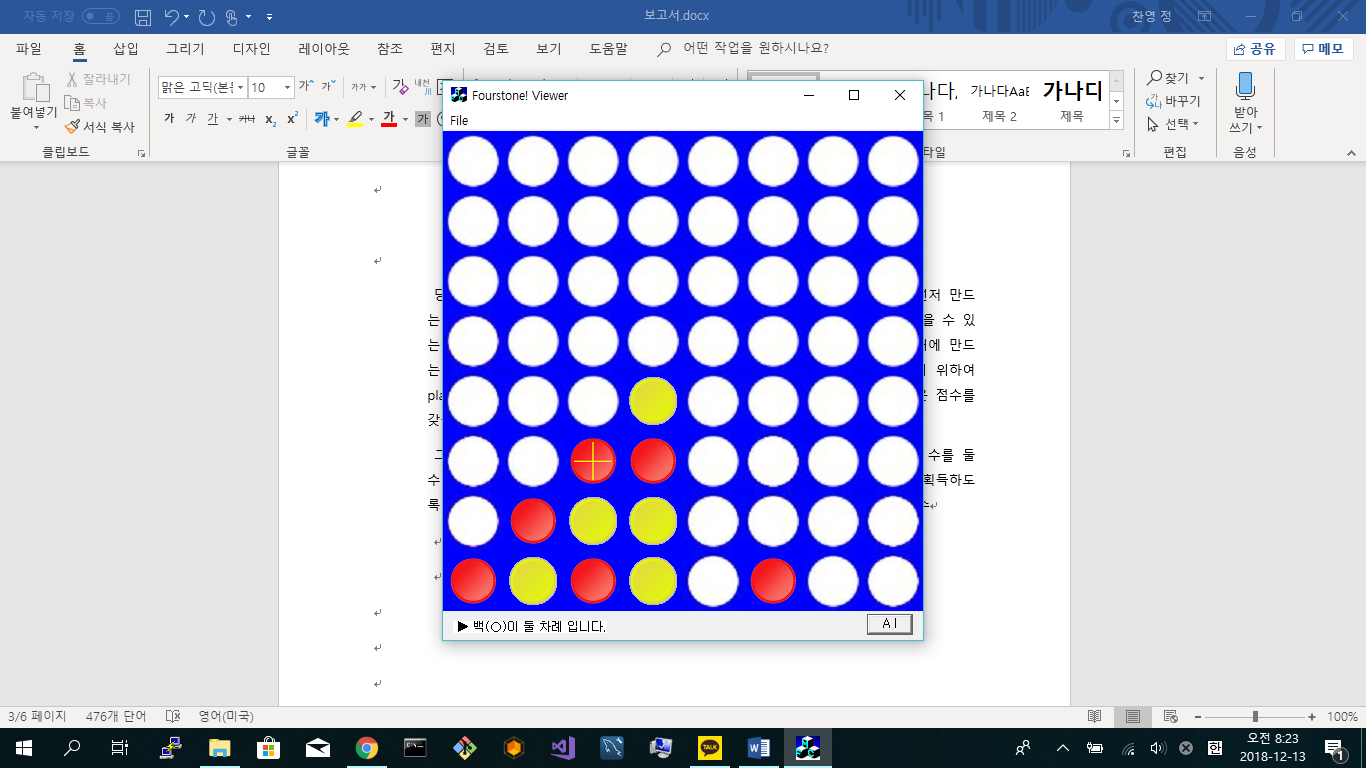
1초안에 둬야 하는 게임 룰상 깊이 5까지만 보았고, 그 과정에서 깊이 0일 때, 즉 무조건 막아야하는 상황일 경우, 깊이 0일 때 점수차이가 많이 크면 그 곳에다가 돌을 놓는 방식으로 진행하였습니다. 소스코드를 보시면 좀 더 자세히 알 수 있습니다.

평가 함수

당연한 이야기이지만 zugzwang을 만들기 위한 최적의 방법은 3개의 연결을 먼저 만드는 것입니다(4. Control of Zugzwang). 3개의 연결을 먼저 만들면 4번째 수를 놓을 수 있는 열에 한하여 승부가 정해지므로 zugzwang을 먼저, 그리고 적의 수보다 아래에 만드는 것이 중요합니다. 따라서 저희는 위의 예시처럼 아래의 수를 먼저 처리하기 위하여 player의 점수에서 opponent의 점수를 빼는 것으로 아래의 수들이 조금 더 높은 점수를 갖을 수 있도록 유도하였습니다.

그리고 단순히 세 개가 연결되는 것이 아니라 세 개가 연결되더라도 네 번째 수를 둘 수 있는지 없는지를 확인하여 다음 수에 대한 포석이 가능한 경우에만 점수를 획득하도록 조건을 설정하였습니다. 이로 인하여 아래와 같은 쓸모 없지만 점수가 높은 수를 두는 경우가 자주 발생하지 않도록 하였습니다.

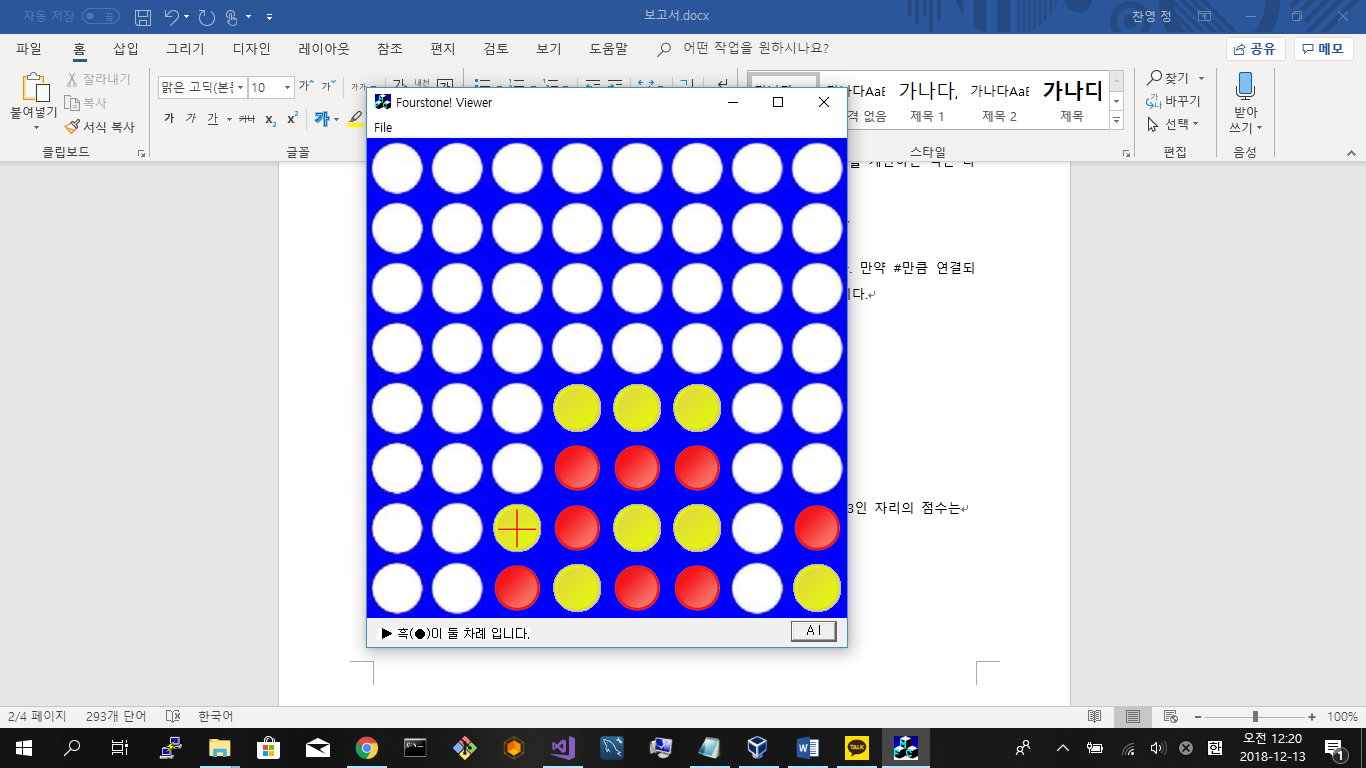
score로서 모든 경우를 환산해서 점수화를 하고, 빨간 돌일떄는 +score 값으로, 노란 돌일 때는 -score 값으로 판단하여 계산 합니다. 모든 열의 점수를 배열로 만들어서 sorting을 합니다. Sorting 후 제일 높은 값을 가지고 있는 열이 선택되고, 그 열에 돌을 두게 됩니다. 만약 점수가 겹치는 경우에는, sorting된 순서에 따라 어떤 열에 돌을 둘 지 정해집니다.



성능 분석

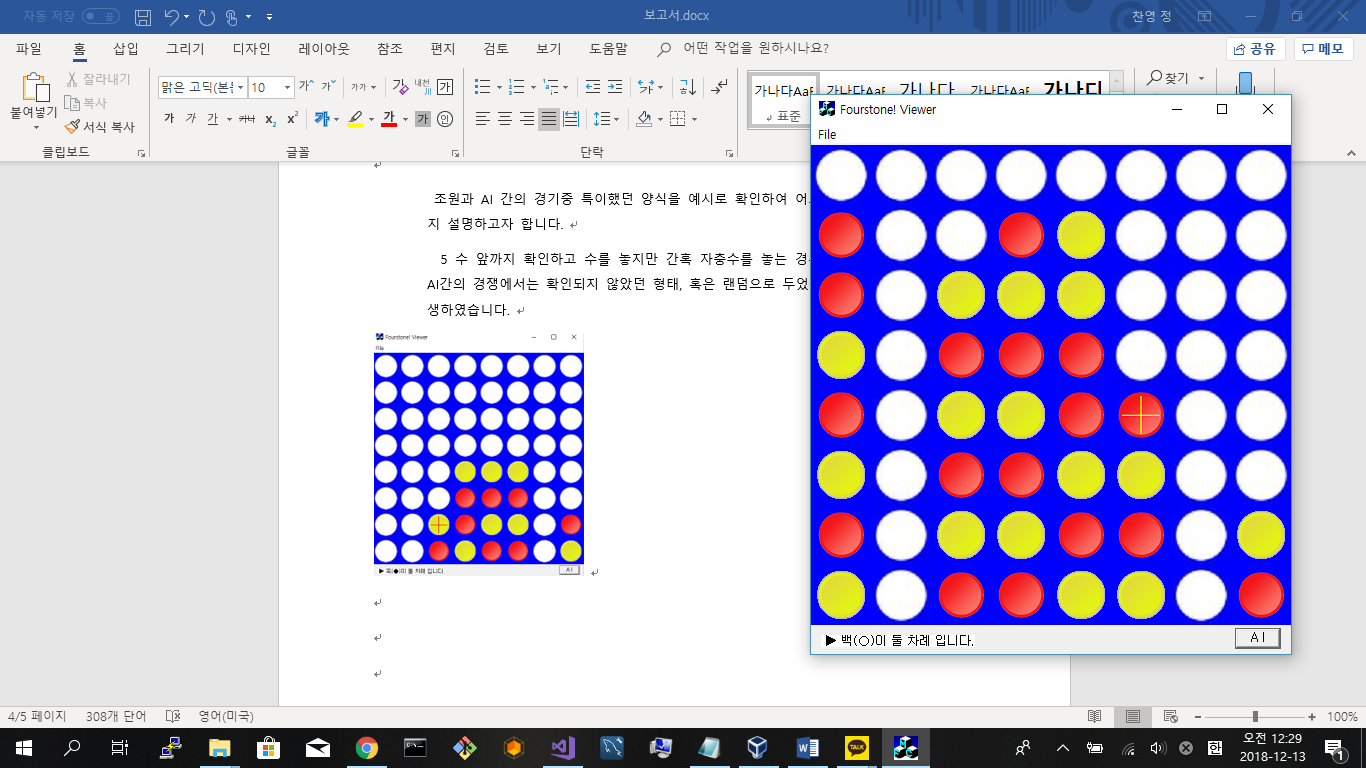
조원과 AI 간의 경기중 특이했던 양식을 예시로 확인하여 어느 부분에서 오류가 있었으며, 어떤 부분에서 더 개선할 수 있는지 확인하기 위하여 특이했던 수들을 모았습니다.

1. 5 수 앞까지 확인하고 수를 놓지만 간혹 자충수를 놓는 경우가 발생하였습니다. 특히, AI간의 경쟁에서는 확인되지 않았던 형태, 혹은 랜덤으로 두었을 때 이런 경우가 자주 발생하였습니다.



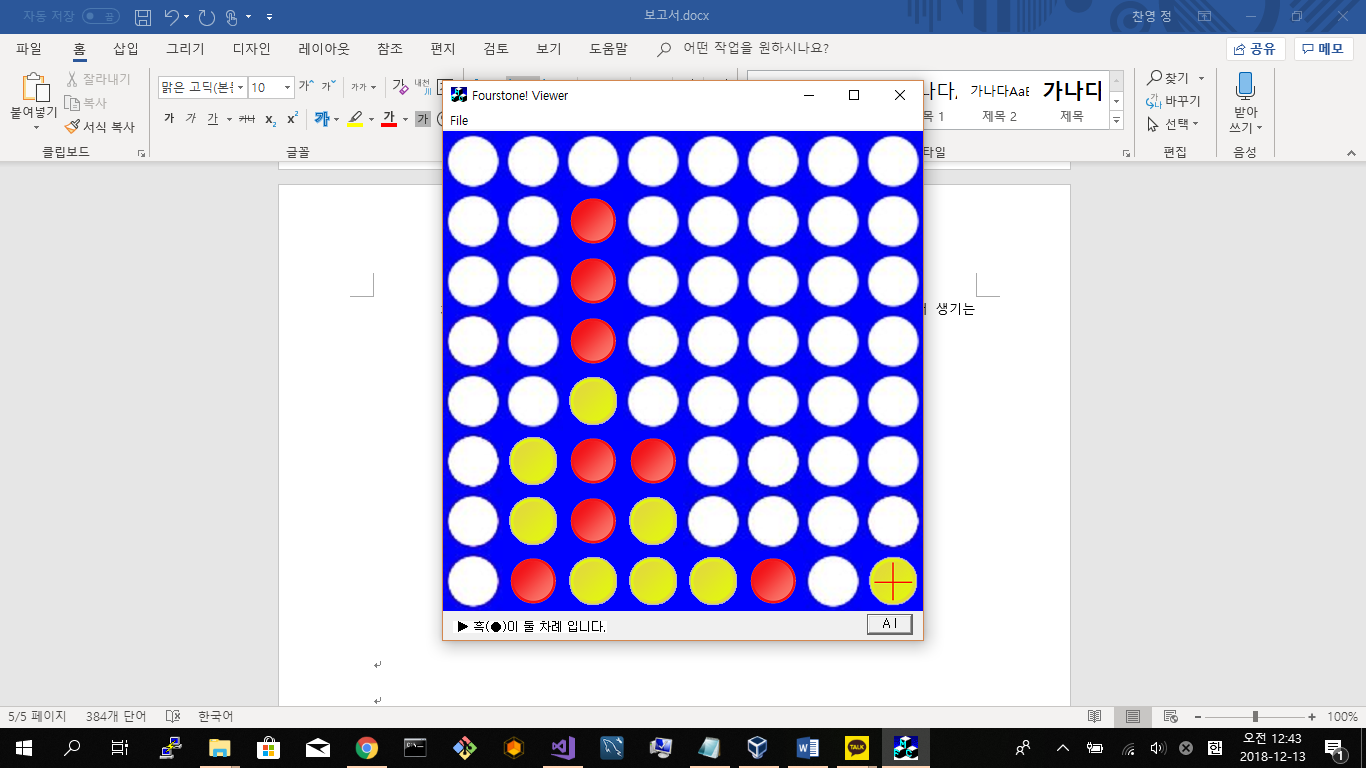
또한, 2번째 열과 7변째 열이 비어 있었을 경우(Zungzwang 이 생기는 경우 등등)에 더욱 빈도가 높게 발생하였습니다.

1. 다른 형태의 자충수는 스스로 ZugZwang을 포기하는 경우가 존재하였습니다.



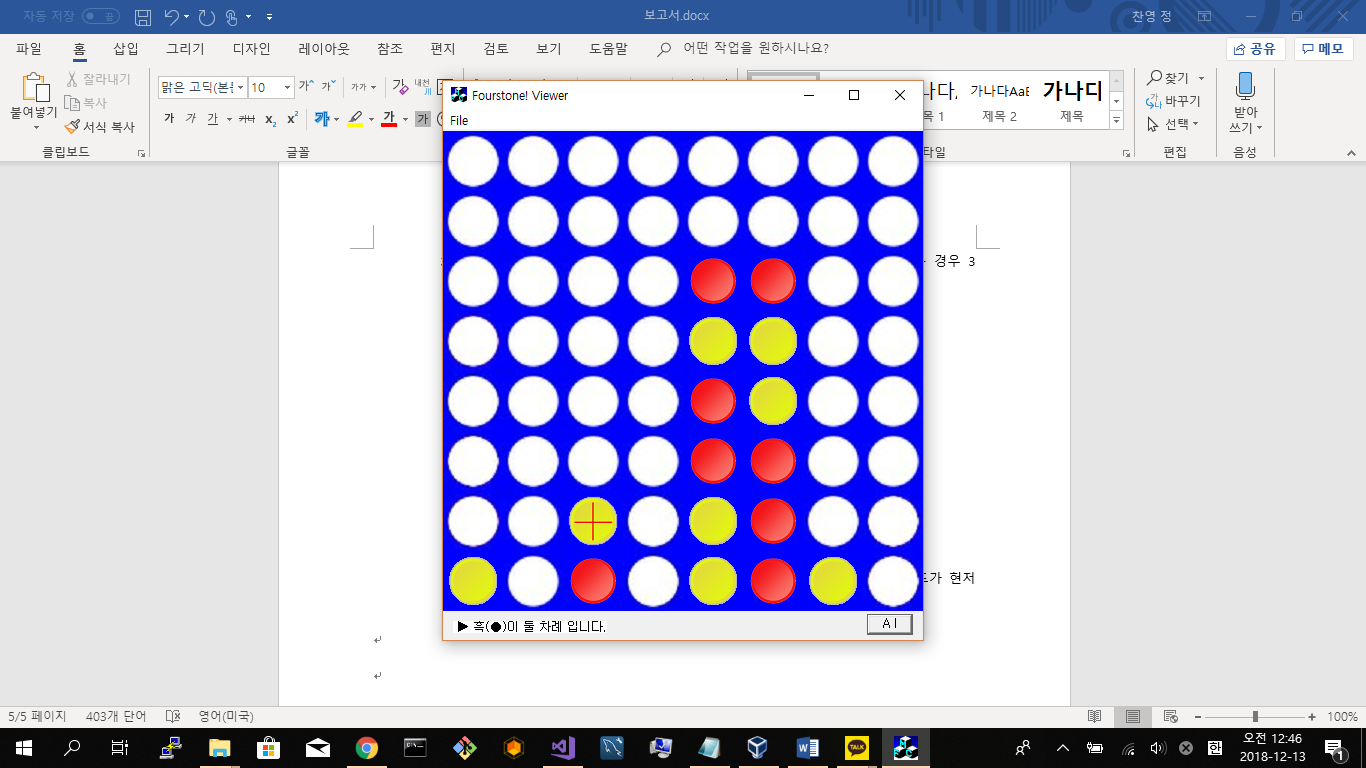
실질적으로는 2열 7열 혹은 8열에서 수를 진행하는 것이 훨씬 좋은 선택이지만 그 열들을 선택하지 않고 6열을 선택하였습니다.

1. 최상단의 행에서 3개가 연결된 경우와 동시에 한 열만 돌출되어서 생기는 경우 3개가 연결된 것을 막지 않는 오류가 발생하였습니다.



기타의 상황에서는 3개의 연결을 거의 무조건 막지만 이 경우 막는 빈도가 현저히 떨어졌습니다.

1. 수를 아래서부터 전개해 나가지 않고 수의 탑을 쌓습니다.



ZugZwang을 만들기 위해서는 아래서부터 수를 만들어 나가는 것이 유리하지만 저희의 AI는 한열을 확정 짓고 다음 수를 풀어나가는 경향을 보였습니다.