BetaGo

- 오목 AI -

홍 길 동

- # 프로젝트 개요
  - ## 개발 배경

    최근 화두가 되었던 이세돌과 구글 AI 프로그램인 알파고의 대결로 인하여 많은 AI와 머신러닝에 집중을 하기 시작하였습니다. 본 프로젝트는 AI 시장 규모가 급증하는 현재 추세에 맞춰 바둑보다 룰도 간편하고 우리에게 더 친숙한 오목에 AI를 이식하여 AI에 친숙해지고자 시작하였습니다.

  - ## 개발 목표

    돌이 연속으로 5개가 놓이면 승리하는 가장 기본적인 오목 알고리즘에 여러 가지 오목 룰 중 대표적인 금지 수인 쌍삼(막힌 곳이 없이 돌이 동시에 3개가 2군데 이상 만들어 지는 수)과 장목(돌이 6개 이상 연속으로 놓이는 경우)을 추가하였고, 여기에 가중치를 이용하여 AI를 추가하였습니다.

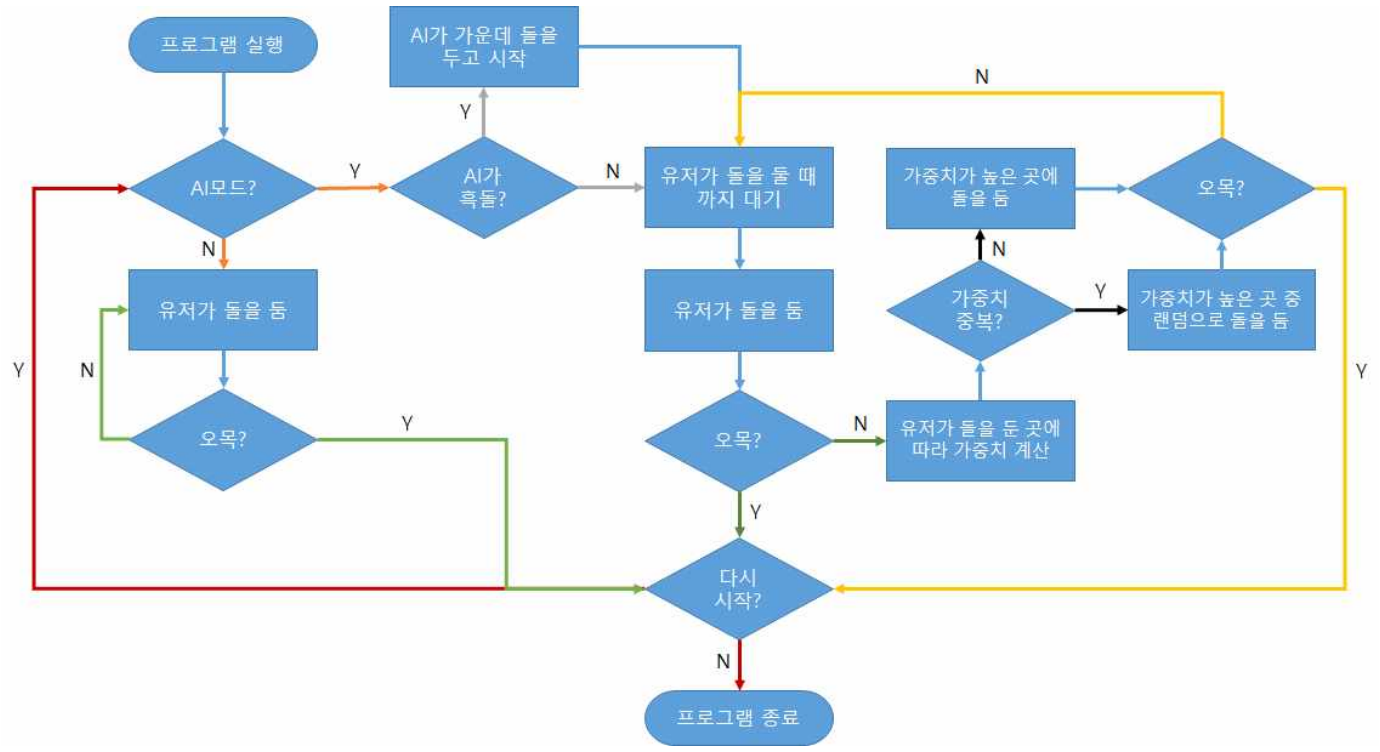
- # 개발 기간

  2016.05.06 ~ 2016.05.20

- # 개발 환경

  구현 언어 : C#
  개발 도구 : Visual Studio 2015
  OS : Window 7

- # 논리적 프로그램 구조



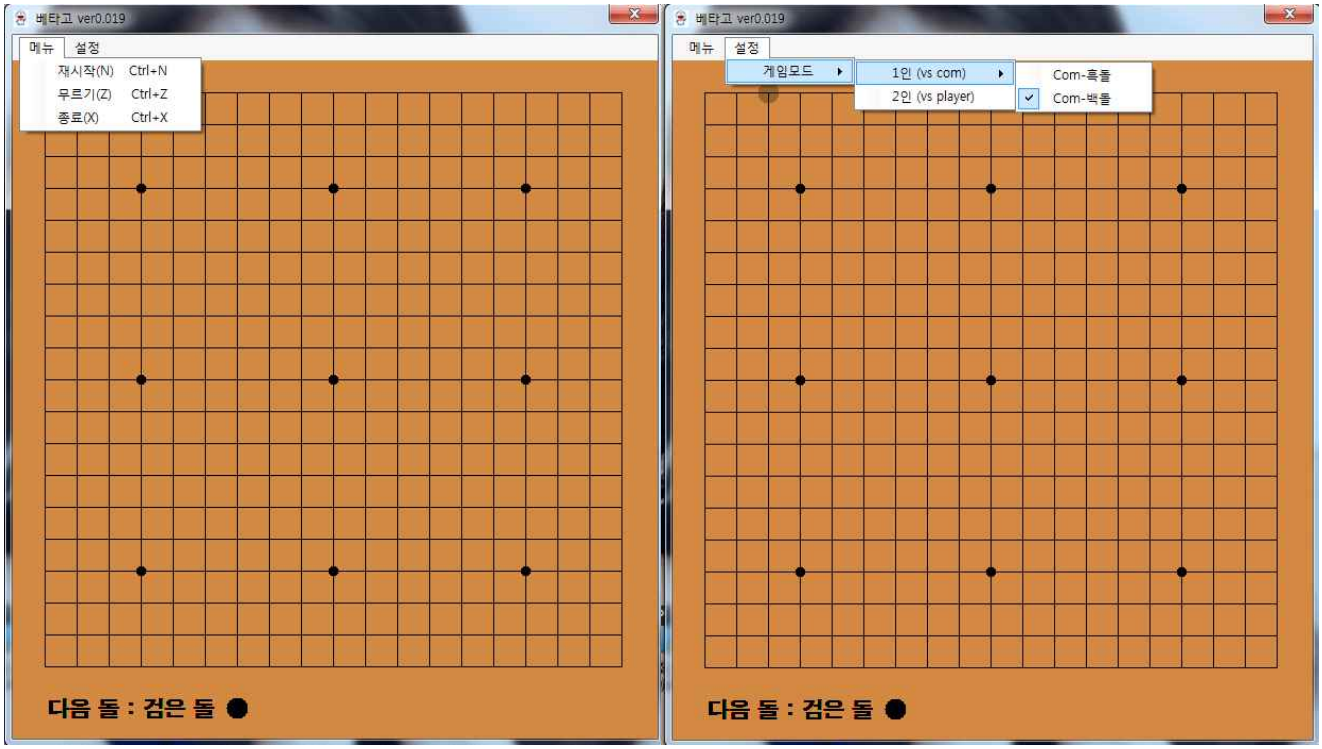
- **프로젝트 상세 설명**
  - **초기 화면 및 기본 설정**

실행 시 “대국이 시작 되었습니다” 라는 음성과 함께 게임이 시작됩니다. 설정은 1인(AI)모드이며, 컴퓨터는 백돌로 설정되었습니다. 바둑판 위에 마우스가 놓이면 마우스 좌표 값을 계산하여 현재 놓여 질 돌의 Argb값에서 A값을 적게 주어서 잔상효과를 구현, 마우스 이동 시 전 좌표를 지우고 현재 좌표를 새로 찍어줍니다.

하단에는 다음에 놓일 돌과 색상을 표기하였습니다.

  - **메뉴 창**

첫 메뉴에는 현재 모드에서 값을 전부 초기화하여 다시 시작하는 재시작, 이번에 돌을 없앨 수 있는 무르기, 게임을 종료하는 종료가 있습니다. 각 메뉴는 옆에 표기된 실행 할 수 있습니다. 설정에는 게임모드를 변경할 수 있는데, 1인과 2인으로 나뉩니다. 1인에서는 컴퓨터의 선, 후공을 정할 수 있도록 흑과 백돌을 설정할 수 있습니다.



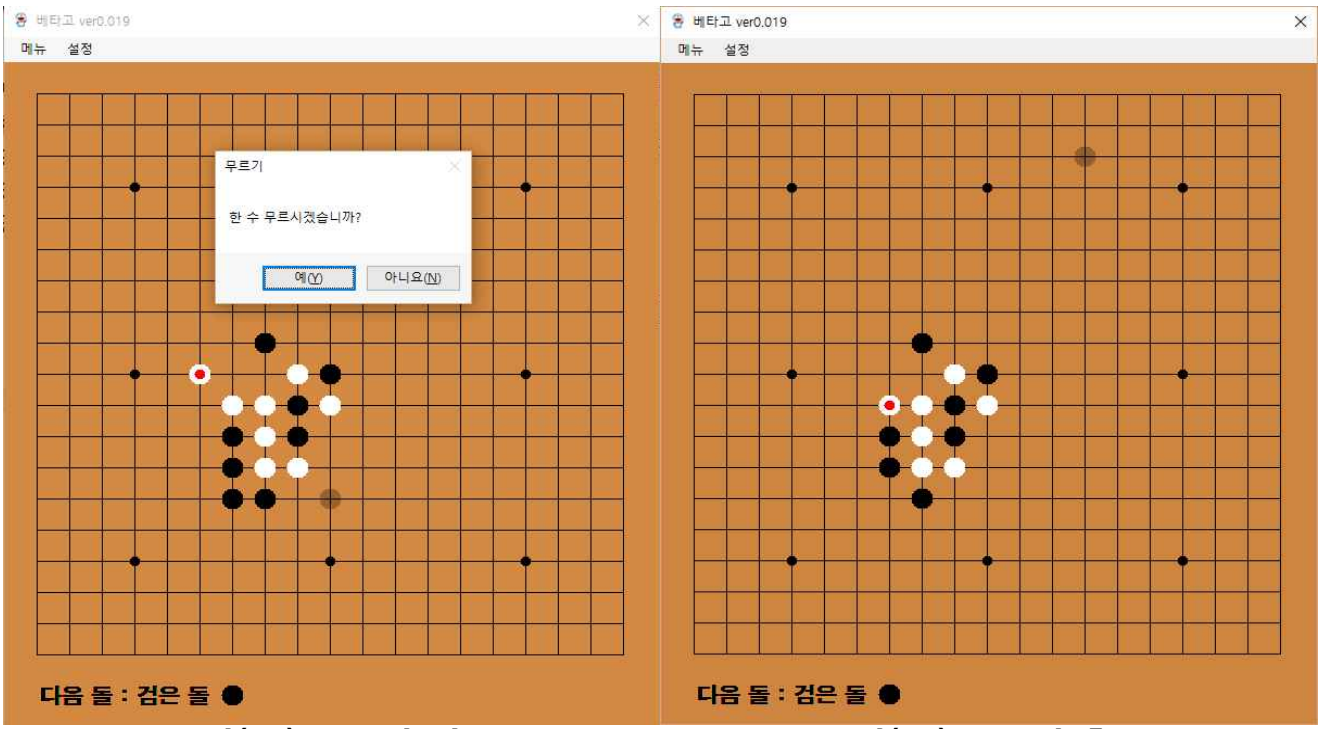
  - **돌 두기**

마우스 왼쪽 버튼 클릭 시 마우스 현재 좌표를 계산하여 바둑판 위에 있을 시 흑돌, 백돌 차례에 따라서 각 색상별로 19x19 배열에 값이 저장되고, 돌이 놓인 배열 인덱스 순대로 스택에 따로 저장하여 무르기 구현 시 순서대로 제거 할 수 있도록 하였습니다.

마지막 돌에 빨간 원을 그리기 위하여 그 돌의 x, y좌표를 따로 저장하여 가장 최근에 어느 곳에 두었는지 쉽게 파악할 수 있도록 구현하였습니다.

-

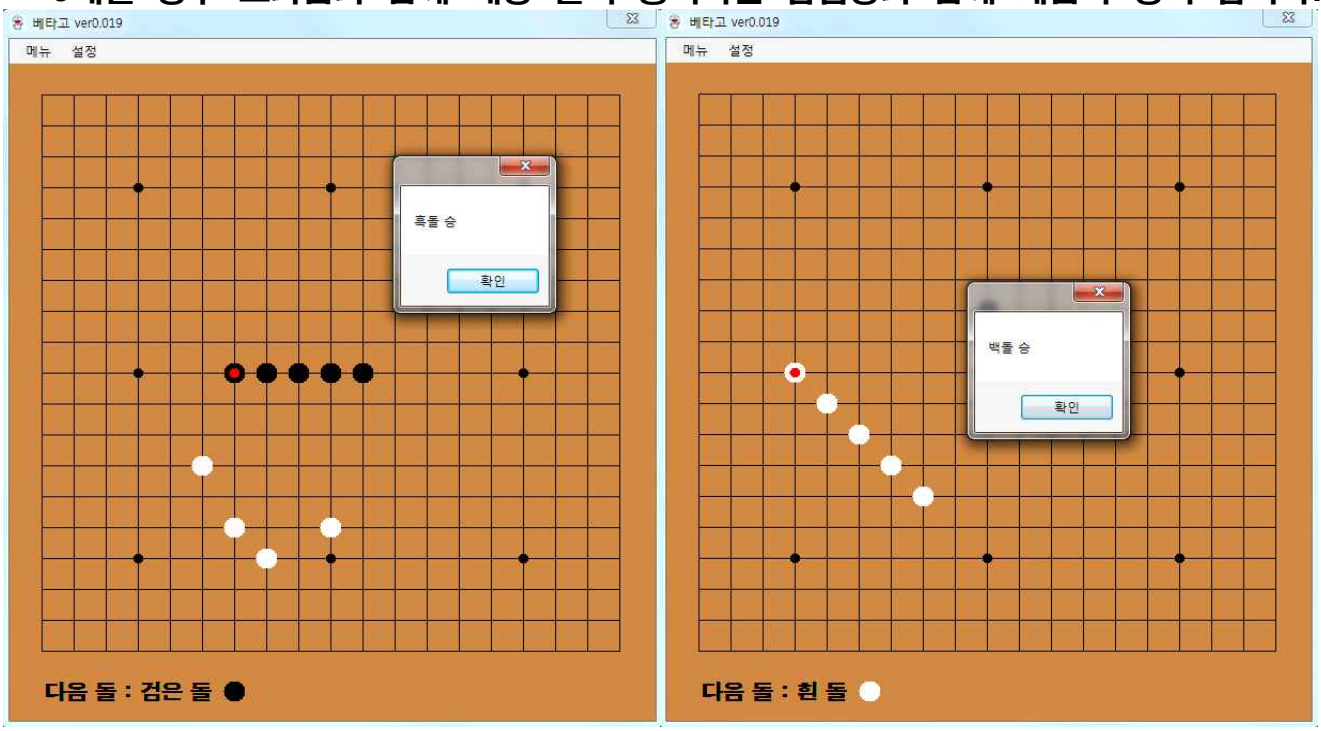
메뉴에서 무르기 또는 단축키(Ctrl + z)입력 시 무르기 요청 음성과 함께 팝업창이 . 예를 클릭하면 스택에 저장되어 있던 돌의 좌표 값 중 맨 위의 값을 제거 한 후 전 좌표의 돌 위에 가장 최근 놓인 돌을 표시하는 빨간 원을 그려준 후 차례를 바꿔줍니다. 1인(AI)모드일 경우 무르기 선택 시 무르기를 두 번 실행 시켜 컴퓨터가 놓인 돌까지 없앨 수 있도록 하였습니다.



<1인(AI) 무르기 전> **<1인(AI) 무르기 후>**

- **승리(오목)**

현재 놓인 돌을 기점으로 상, 하, 좌, 우, 대각선을 검사하여 연속된 돌의 개수가 5개일 경우 효과음과 함께 해당 돌의 승이라는 팝업창과 함께 게임이 정지 합니다.



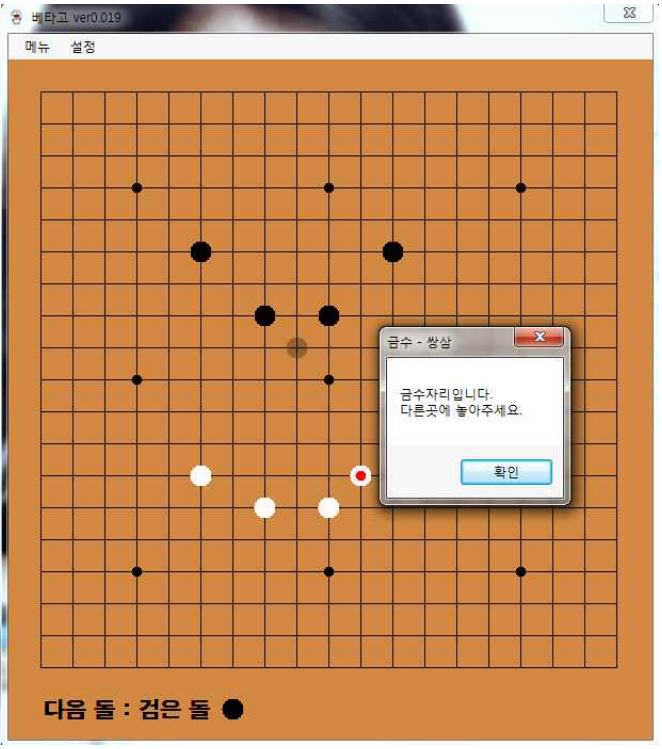
<흑돌 가로 승리> **<백돌 역사선 승리>**

- **금수(쌍삼, 장목)**

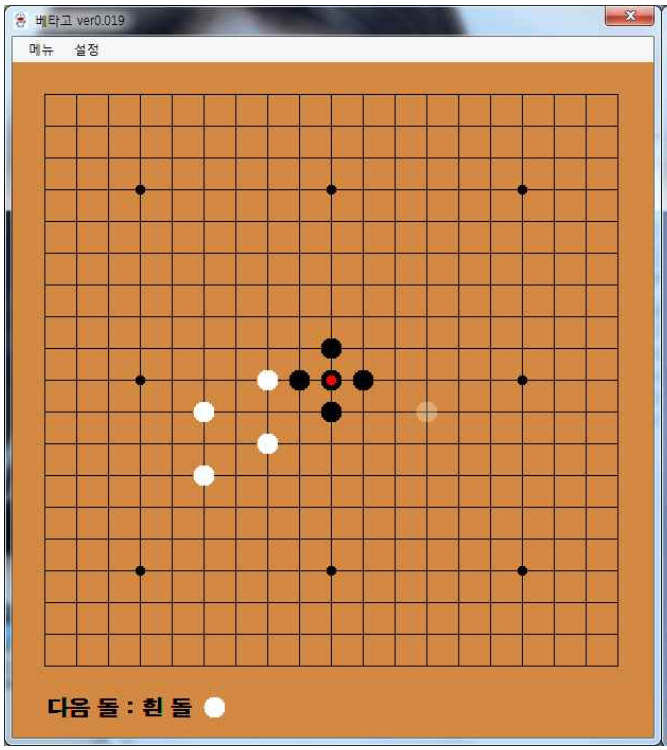
차례일 때 돌이 놓이는 순간 가로, 세로, 대각선 전부 확인 하여 벽이나 다른 색의 돌이 막고 있지 않는 상태이고, 같은 색의 돌이 연속으로 3개 또는 빈칸 한곳 포함 3개가 2군데 이상 만들어 진다면 팝업창으로 경고를 한 후 돌을 놓인 자리를 초기화합니다.

장목의 경우 오목이 완성되었는지 확인하는 함수에서 돌이 놓인 위치로부터 상, 하, , 우, 대각선, 전부 확인하여 정확히 5개가 연속으로 놓인 경우에만 승리하도록 구현하였습니다.

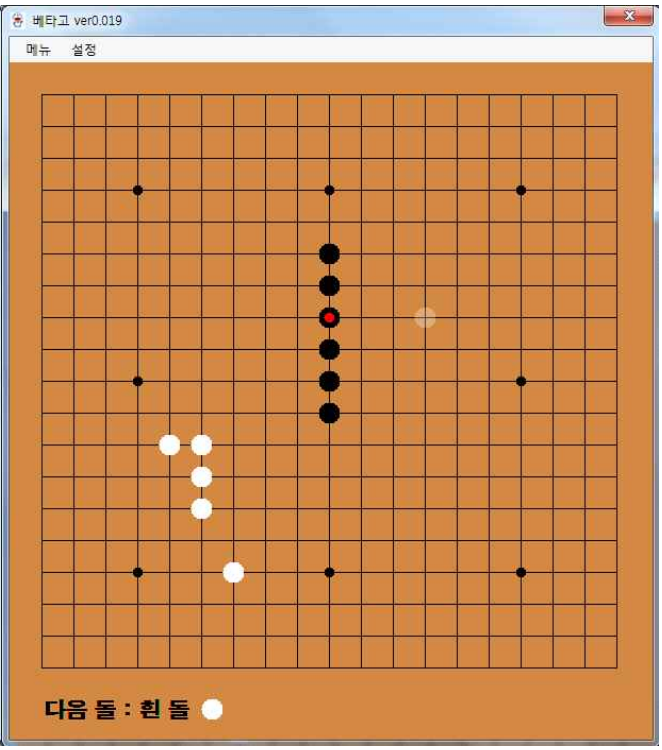
<금수 – 쌍삼 확인>



<금수 – 쌍삼(빈칸포함) 확인>



<쌍삼 막혔을 경우>



<육목(장목)일 경우>

- **AI(가중치)**

바둑알 색을 저장한 후 바둑판의 처음 인덱스부터 마지막 인덱스까지 각 인덱스를 기준으로 현재 놓인 자신의 돌과 상대의 돌을 토대로 경우에 따라서 해당 인덱스에 가중치를 저장한다. 저장된 가중치중 가장 큰 값에 자신의 돌을 두게 됩니다. 만약 가중치가 될 경우 난수를 생성하여 중복된 가중치 중 랜덤으로 한 곳에 두도록 구현하였습니다. 컴퓨터가 선공(흑돌)일 경우에는 컴퓨터가 먼저 돌을 두어야 하기 때문에 만약 바둑판에 돌이 하나도 없을 경우 중앙에 놓도록 구현하였습니다.

```
enum POINT // 각 상황에따른 점수판의 점수 정의
   {
    P_GOOD0 = 1,                        // 돌이 없는칸
    P_BADb1 = 1250, P_BAD1 = 5000,      // 한쪽이 막혀있는 적의 돌, 양쪽이 뚤려있는 적의 돌
    P_GOODb1 = 1250, P_GOOD1 = 5000,    // 한쪽이 막혀있는 나의 돌, 양쪽이 뚤려있는 나의 돌
    P_BADb2 = 2500, P_BAD2 = 12000,     // 한쪽이 막혀있는 2개의연속된 적의 돌, 양쪽이 뚤려있는 2개의연속된 적의 돌
    P_GOODb2 = 2500, P_GOOD2 = 12000,   // 한쪽이 막혀있는 2개의연속된 나의 돌, 양쪽이 뚤려있는 2개의연속된 나의 돌
    P_BADb3 = 11000, P_BAD3 = 60000,    // 한쪽이 막혀있는 3개의연속된 적의 돌, 양쪽이 뚤려있는 3개의연속된 적의 돌
    P_GOODb3 = 11000, P_GOOD3 = 130000, // 한쪽이 막혀있는 3개의연속된 나의 돌, 양쪽이 뚤려있는 3개의연속된 나의 돌
    P_BADb4 = 200000, P_BAD4 = 200000,  // 한쪽이 막혀있는 4개의연속된 적의 돌, 양쪽이 뚤려있는 4개의연속된 적의 돌
    P_GOOD4 = 99999999                  // 4개연속인 나의 돌
};

/* // 개선 (공격적으로 변경)
enum POINT // 각 상황에따른 점수판의 점수 정의
{
    P_GOOD0 = 1,                        // 돌이 없는칸
    P_BADb1 = 500, P_BAD1 = 2000,       // 한쪽이 막혀있는 적의 돌, 양쪽이 뚤려있는 적의 돌
    P_GOODb1 =1250, P_GOOD1= 5000,      // 한쪽이 막혀있는 나의 돌, 양쪽이 뚤려있는 나의 돌
    P_BADb2 = 1000, P_BAD2= 4000,       // 한쪽이 막혀있는 2개의연속된 적의 돌, 양쪽이 뚤려있는 2개의연속된 적의 돌
    P_GOODb2 = 2500, P_GOOD2 = 12000,   // 한쪽이 막혀있는 2개의연속된 나의 돌, 양쪽이 뚤려있는 2개의연속된 나의 돌
    P_BADb3 = 11000, P_BAD3 = 60000,    // 한쪽이 막혀있는 3개의연속된 적의 돌, 양쪽이 뚤려있는 3개의연속된 적의 돌
    P_GOODb3 = 11000, P_GOOD3 = 130000, // 한쪽이 막혀있는 3개의연속된 나의 돌, 양쪽이 뚤려있는 3개의연속된 나의 돌
    P_BADb4 = 200000, P_BAD4 = 200000,  // 한쪽이 막혀있는 4개의연속된 적의 돌, 양쪽이 뚤려있는 4개의연속된 적의 돌
    P_GOOD4 = 99999999                  // 4개연속인 나의 돌
};*/
```

<상황별 가중치 설정>