2016년도 1학기

동국대학교 심화 프로그래밍(CES2022)

2016.05.12. 조교 송치원

0 우리가 실습으로 할 것



1 우리가 아는 '지뢰 찾기' 게임



목표

 지뢰를 피하면서 빈 사각형을 찾는 것이 이 게임의 목 표입니다. 보드에서 모든 빈 사각형을 빨리 찾을수록 점수가 높습니다.

게임 방법

- 지뢰 찾기의 규칙은 간단합니다.
- 지뢰를 클릭하면 게임이 끝납니다.
- 빈 사각형을 클릭하면 게임을 계속할 수 있습니다.
- 숫자가 나타나면 그 숫자를 통해 주위의 8개 사각형에 숨겨진 지뢰 수를 알아낼 수 있습니다. 이 정보를 사용하여 주위의 사각형 중 어떤 사각형이 클릭했을 때 안전한지 추측해야 합니다.

1 우리가 아는 '지뢰 찾기' 게임

• 숫자의 의미: 현 위치에서 인접한 8방향에 존재하는 지뢰 개수

1	1	1	0
1	*	1	0
1	1	1	0
0	0	0	0

*	*	*	2
*	8	*	3
*	*	*	2
2	3	2	1

*	1	0	0
1	1	0	0
0	0	0	0
0	0	0	0

*	1	1	*
1	1	1	1
1	1	1	1
*	1	1	*

- 숫자가 높을수록 지뢰가 주변에 지뢰가 많이 존재한다.
- 0의 경우는 인접한 8방향에 지뢰가 존재하지 않는다.
- 8의 경우 인접한 8방향에 모두 지뢰가 존재한다.

2 우리가 할 '지뢰 찾기' 게임

- 목표 : 하나의 게임 보드에서 지뢰를 많이 찾자!
- 총 플레이어의 수 : 2명
- 플레이 방식 : 턴
- 한 턴에 이루어지는 행동
 - 1. (x, y) 위치의 값을 확인한다.
 - 2. 지뢰가 존재할 경우
 - ① 현재 턴의 플레이어의 점수를 올린다.
 - ② (x, y) 위치를 두 플레이어에게 알린다.
 - ③ 현재 턴의 플레이어가 다시 1번부터 한다.
 - 3. 지뢰가 존재하지 않을 경우
 - ① 현재 위치에 어떤 값이 있는지 두 플레이어에게 알린다.
 - ② 다음 턴의 유저가 1번부터 시작한다.

3 무엇을 해야 하는 걸까요?

- 지뢰 찾기 프로그램의 전체적인 틀은 구현되어있습니다.
- 앞에서의 룰에 따라 플레이 했을 때 상대방을 이길 수 있는 Logic 구현
- 본인의 생각을 구현해내는 능력을 길러봅시다.
- 간단한(심플한) AI를 구현하여 챔피언에 도전해보세요.
- 실제로 제출한 과제로 대회가 치러집니다.

4 코드를 알아보기에 앞서

- 전체 프로그램 코드를 이해해야만 할 수 있는 과제가 절대 아닙니다.
- 전체를 몰라도 어떤 기능을 구현해야 하는지를 명확히 알면 됩니다.
- 실제로 게임을 한다면 어떤 칸을 열어볼지에 대해서 잘 고려해보세요.
- 본인이 정리되지 않은 것을 코드로 옮기긴 어렵습니다.
- 어떻게 하면 '조금 더' 효율적일지 생각해보세요.

Common.h

- #define MINE_COUNT 100
 - 폭탄의 개수
- #define BOARD_SIZE 200
 - 보드의 크기

Point.h

- Point(int,int)
 - x, y좌표를 매개변수로 받아서 바로 값을 설정하는 생성자
- int getX();
- int getY();
 - x, y좌표를 각자 반환함
- Point& operator = (const Point &p);
- bool operator == (const Point &p);
- bool operator != (const Point &p);
 - 연산자 오버라이딩
- bool checkBoardRange();
 - 설정된 x, y좌표가 보드 범위 이내인지 아닌지 판별

Game,h

- int board[BOARD_SIZE][BOARD_SIZE]
 - 실제 게임 보드

- void insertMine(Point p)
 - 지뢰를 매복시키는 메소드
- int getBoardInfo(Point p)
 - 특정 위치에 있는 값을 반환

Player.h

- virtual Point input()
 - 확인해보고 싶은 위치를 반환
- virtual void checkMineInfo(Point p)
 - 확인한 위치에 지뢰가 있을 경우 호출될 메소드
 - 지뢰 위치 p
- virtual void checkBoardInfo(Point p, int value)
 - 확인한 위치에 지뢰가 없을 경우 호출될 메소드
 - 지뢰 위치 p, 인접 8방향에 존재하는 지뢰의 수 value
- 해당 class를 상속받아서 내부적으로 알아서 구현하면 됩니다.

GameManager

- Assistant player1;
- M2012345678 player2;
 - 게임을 플레이할 player1과 player2
 - Class를 변경하여 본인이 구현한 코드의 성능을 비교하면 된다.
- Game game;
 - 실제 게임 보드
- int player1Score;
- int player2Score;
 - 플레이어들의 점수를 저장할 변수
- void play();
 - 게임을 플레이할 메소드

주의사항

- 입출력을 절대로 하지마세요.
 - cin, cout, scanf, printf 등 모든 입출력을 금지합니다.
- 클래스 명과 cpp파일 명과 header 파일명은 "M학번" 으로 합니다.
 - Ex) 학번이 2012345678일 경우 M2012345678
- 반드시 header 파일을 생성하여 class를 선언해주어야 합니다.
- cpp파일에는 본인이 구현한 class만이 존재하도록 합니다.

과제 일정

- 5월 12일 : 과제 출제
- 5월 19일 : 중간 확인
 - 조교의 AI와 대결
 - 만약 조교의 AI에게 이기면 이번 과제에 대해 만점!
- 5월 26일 : 최종 과제 마감
 - 1분 내외로 본인의 방법을 스피치
 - Ppt 준비하지 마세요. 간략하게 "이런 생각으로 했다."를 전달해주세요.
 - 실제 제출 코드를 활용하여 대결 진행

주석과 보고서

- 주석은 절대 Line마다 달지 않는다.
- 메소드 위에 어떤 생각으로 구현한 것인지를 간략하게 적는다.
 - 해당 주석만 보고 어떤 식으로 구현된 것인지 이해 가능하도록.

```
/**
이러이러해서 저러저러해서
이러니 저러니 한 방법으로 함.
*/
```

• 보고서는 각 메소드별로 위에서 작성한 주석을 그대로 적는다.

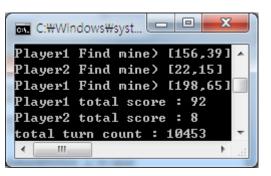
Point M2012345678::input() 러이러해서 저러저러해서 이러니 저러니 한 방법으로 함.

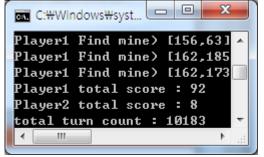
- 추가로 본인의 AI와 랜덤 AI의 대결 결과 6개를 캡처하여 붙인다.
 - 본인이 찾은 개수의 평균도 기입한다.

조교 vs 랜덤의 결과 내용

```
Player1 Find mine | [167,6] | Player2 Find mine | [156,53] | Player1 Find mine | [184,33] | Player1 total score : 91 | Player2 total score : 9 | total turn count : 11076 | |
```

```
Player1 Find mine) [172,110 A
Player1 Find mine) [185,68]
Player2 Find mine) [76,109]
Player1 total score : 84
Player2 total score : 16
total turn count : 11095
```





- 조교(Player1)와 랜덤(Player2)의 4번의 경기
- 약 10,000~11,000번 정도의 확인 작업을 한다.
 - 랜덤의 경우 이미 확인한 곳도 중복으로 확인함
- 랜덤의 성능이 생각보다 나쁘지 않다.
 - 실제로 1~2개 정도의 지뢰 찾기를 생각했다.
 - 보드의 크기가 더 커진다면 성능은 당연히 떨어질 것이다.
- · 조교의 방법은 대부분 90개 정도의 지뢰를 찾아냈다.
 - 굉장히 대충 구현을 한 것이다.