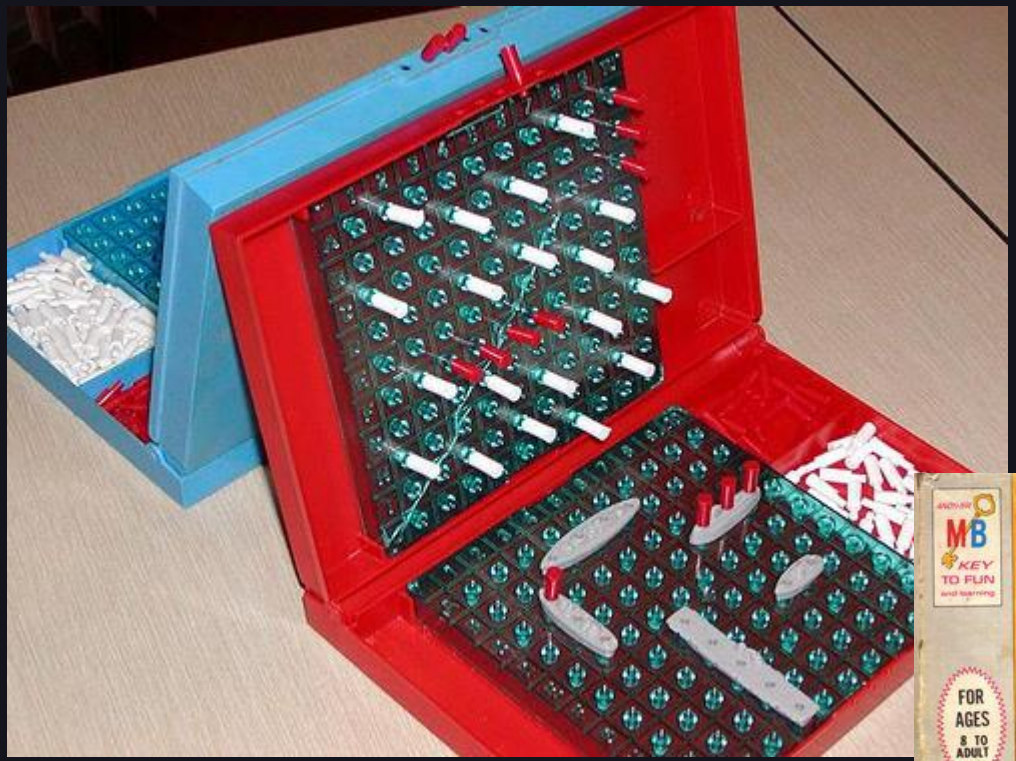
C++ BattleShip 프로젝트

국민대학교 소프트웨어 학부
C++ 배틀십(BattleShip) 프로젝트

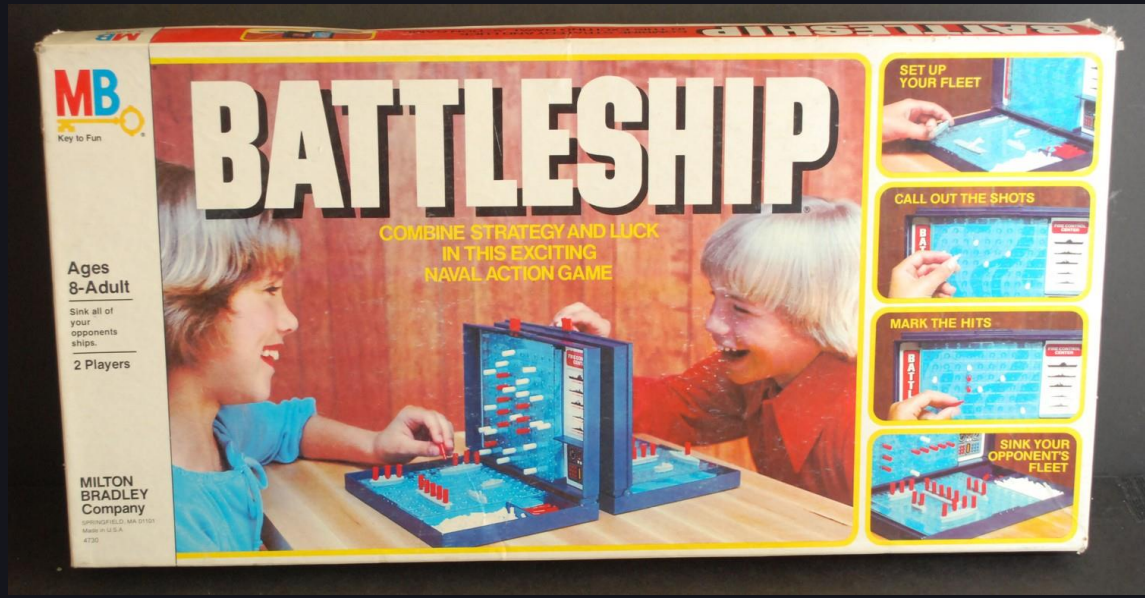
박민근
(agebreak@kookmin.ac.kr)

BattleShip Game




MB
KEY TO FUN
AMERICA'S ALL TIME FAVORITE GAME
BATTLESHIP
IT'S A HIT
G-4
FOR AGES 8 TO ADULT
MILTON BRADLEY COMPANY
SPRINGFIELD MASSACHUSETTS
4730
MADE IN U.S.A.
BATTLESHIP GAME


BATTLESHIP
MB
BATTLESHIP
THE CLASSIC
NAVAL COMBAT GAME


MB
Key to Fun
BATTLESHIP
COMBINE STRATEGY AND LUCK
IN THIS EXCITING
NAVAL ACTION GAME
Ages
8-Adult
Sink all of your opponents ships.
2 Players
MILTON
BRADLEY
Company
SET UP
YOUR FLEET
CALL OUT THE SHOTS
MARK THE HITS
SINK YOUR
OPPONENT'S
FLEET

BattleShip Game

=숫자야구의 2차원 버전

Battleships!

My Ships

A								
B								
C								
D								
E								
F								
G								
H								
	1	2	3	4	5	6	7	8

Aircraft Carrier | A | A | A | A | A |

Battleship | B | B | B | B |

Cruiser | C | C | C |

Destroyers | D | D | | D | D |

Their Ships

A								
B								
C								
D								
E								
F								
G								
H								
	1	2	3	4	5	6	7	8

Aircraft Carrier | A | A | A | A | A |

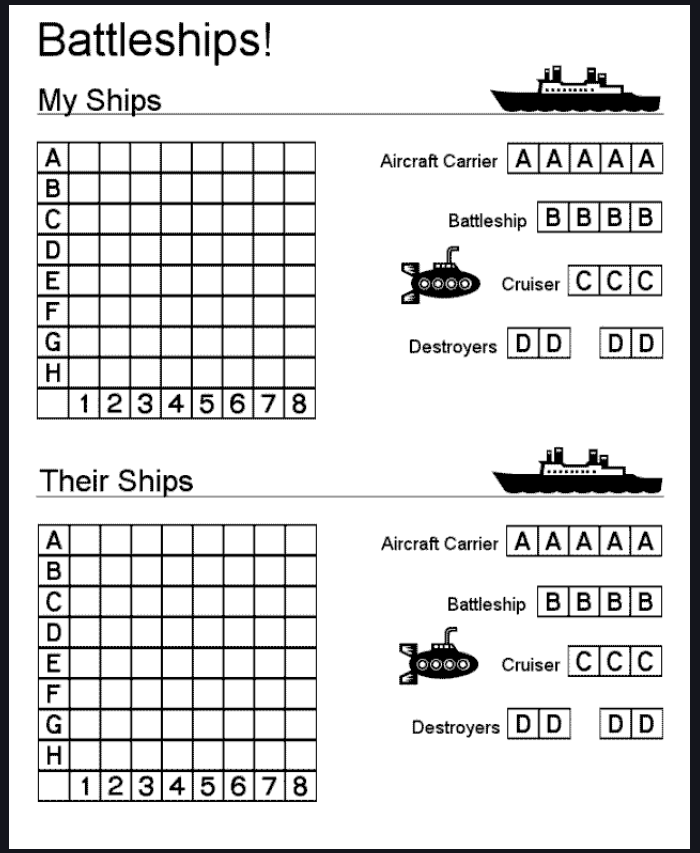
Battleship | B | B | B | B |

Cruiser | C | C | C |

Destroyers | D | D | | D | D |

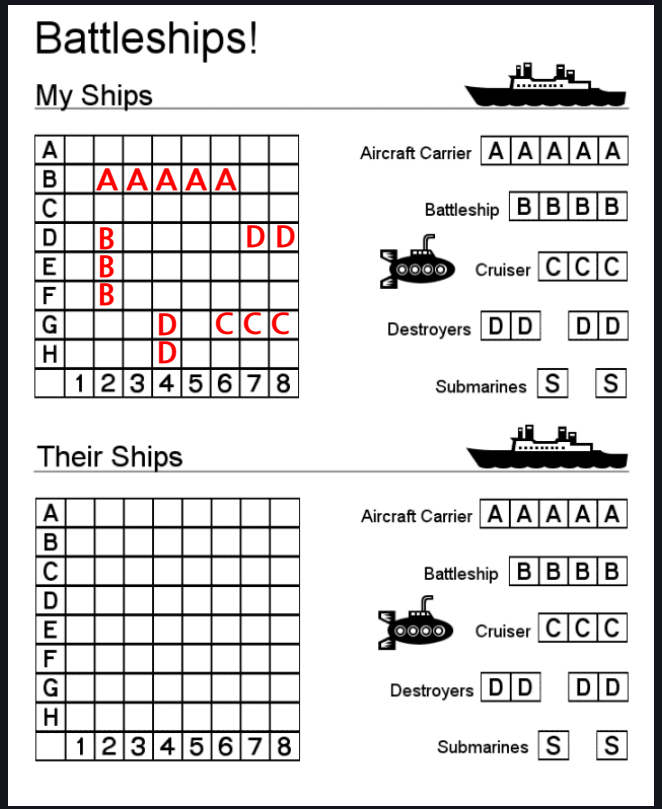
BattleShip Game Rule (1)

1. 게임을 시작한다
2. 두명의 플레이어는 오른쪽 게임판을 가지고 게임을 시작 한다.



BattleShip Game Rule (2)

1. 각자 My Ships 맵에 자신의 배들을 랜덤으로 배치 한다.
2. 임의의 위치에 가로 or 세로 배치 한다 (대각선은 안됨)
    1. 에어크래프트 : 5칸 X 1개
    2. 배틀십 : 4칸 X 1개
    3. 크루저 : 3칸 X 1개
    4. 디스트로이어 : 2칸 X 2개
    5. 서브마린 : 없음 (그림에는 있지만, 제외)

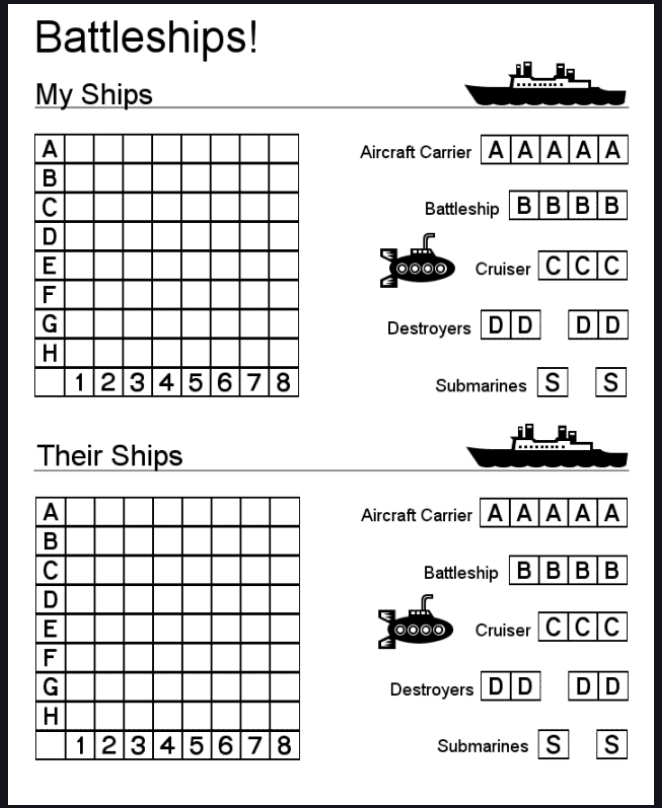


BattleShip Game Rule (3)

1. 맵에는 가로 혹은, 세로로 배치가 가능하다.
2. 같은 자리에 중첩되게 배치하지 못한다.
3. 하나의 배는 분리되서 배치되지 못한다.

   (ex. 에어크래프트는 가로 혹은 세로 직선으로 5칸을 차지한다)

4. 배의 일부가 맵 밖으로 나가서는 안된다.

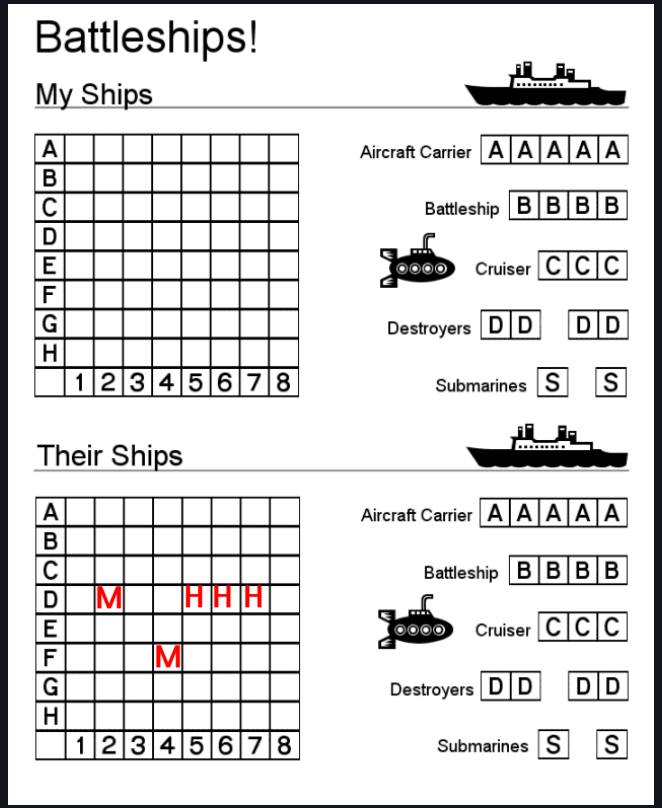


BattleShip Game Rule (4)

1. 한턴씩 교대로 Attaker와 Defneder가 된다
2. Defender가 배치를 완료하면, Attacker는 매턴마다 맵의 임의의 위치를 공격한다. (ex. A2, G6, F4…)
3. Defender는 공격받은 맵 위치를 체크하여 결과를 알려 준다.

BattleShip Game Rule (5)

1. 공격 결과로 Defender는 아래 중의 하나의 결과를 반환한다.
    1. MISS : 공격 위치에 아무것도 없는 경우
    2. HIT : 공격 위치에 배가 있는 경우 (맞은 배의 종류는 알려주지 않는다)
    3. DESTORY : 이 공격으로 인해서 배가 침몰한 경우. 어떤 배가 침몰했는지 알려준다
2. 공격자를 결과를 자신의 맵에 체크하여, 다음 공격 위치를 결정할때 참고 한다



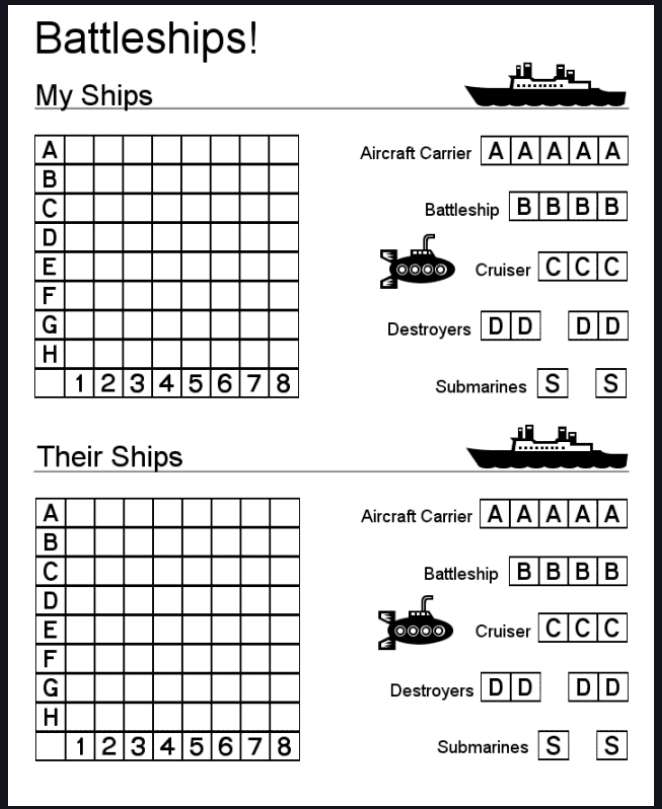
BattleShip Game Rule (6)

1. 침몰
    1. 배의 모든 영역이 HIT되면 침몰된다.
    2. 마지막 영역까지 HIT되어 침몰될 때는 HIT가 아닌 DESTROY를 반환하고, 배의 종류를 알려줘야 한다.

Ex. 크루저의 첫번째 칸 피격 - HIT

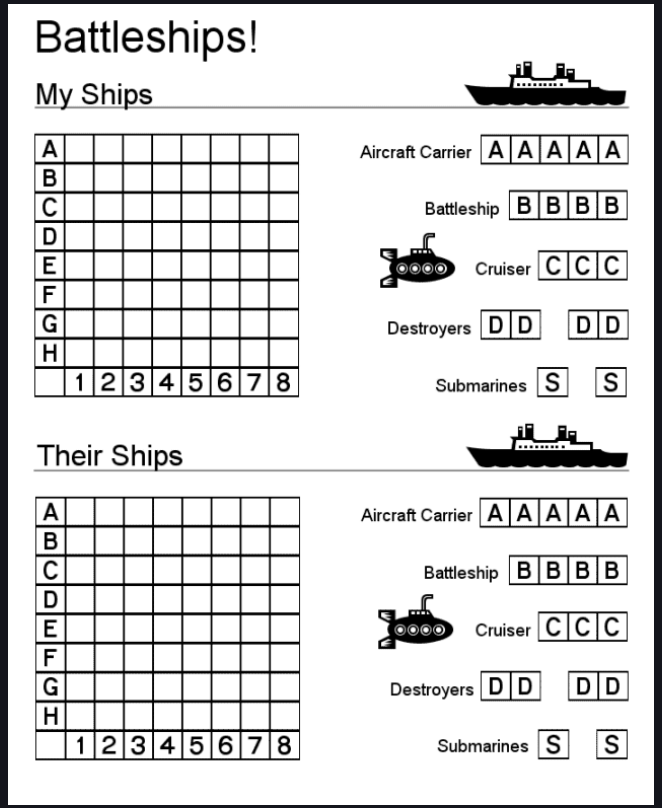
세번째 칸 피격 - HIT

두번째 칸 피격 - DESTROY:크루저



BattleShip Game Rule (7)

1. 게임의 종료
    1. 교대로 한번씩 좌표를 불러서, 상대방의 배의 종류를 모두 침몰 시키면 승리 한다.



BattleShip Game
= C++ 프로젝트 버전

BattleShip Project #1

1. 턴제 게임이다.
2. 1명은 공격자(Attacker)가 되고, 1명은 방어자(Defender)가 된다.
3. 원래 룰은 1턴씩 공격<->방어 이지만, 여기서는 공격자(유저)와 방어자(컴퓨터)를 따로 정의하고, 방어자는 배를 배치만 하고, 공격자만 공격 한다.

BattleShip Project #2

1. 1단계 프로젝트
    1. 방어자는 자동으로 배를 랜덤으로 배치한다.
    2. 플레이어는 공격자가 되어서, 좌표를 입력하여 상대방의 배를 격추하는 형태로 제작 한다.
    3. 매 턴 좌표를 입력하고, 모든 배를 격추하면 종료 된다.
2. 2단계 프로젝트
    1. 사용자의 개입은 없이, 자동으로 AI가 게임을 진행하도록 제작한다.
    2. 랜덤으로 좌표를 매 턴 불러서 상대방의 모든 배를 격추 되면 종료 되도록 한다.
    3. 10 게임을 반복 실행하고, 걸린 평균 턴 수를 출력한다.

BattleShip Project #3

1. 1턴 : Attacker가 공격 -> Defender가 결과를 알려준다 -> 결과에 대한 판단 및 처리
2. Attacker는 프로그래밍된 알고리즘을 이용하여 매 턴 공격을 반복한다.
3. 위 턴을 반복하면서, Defender의 모든 배를 격추하면, 한 게임이 종료 된다.
4. 총 10게임을 자동으로 수행하여, 최종적으로 게임당 평균 턴 수를 출력한다.

```
// 의사 코드
While(IsAllDestory())
{
   ++turn;
   hitResult = 공격(좌표);
   CheckHitResult(hitResult);
}
```

BattleShip Project #4

1. **2 -1 단계 (+ 점수)**
    1. 랜덤 배치로는 턴 수가 너무나 걸린다. (최악의 경우 8X8 = 64턴)
    2. 인간이 생각하는 방법을 참고해서, 효율적인 AI 알고리즘을 구현한다.
    3. 힌트 : 타겟에 한번 히트했는데, 다음 턴에 완전히 다른 좌표를 부를까??
2. **3단계 프로젝트 (+ 점수)**
    1. 위의 간단한 AI 보다 더 효율적인 AI 알고리즘을 구현할 수 없을까?
    2. 힌트 : 한번도 공격하지 않은 빈맵에서 맨 구석자리가 맞을 가능성이 높을까? 정중앙이 맞을 가능성이 높을까?
3. **위의 스마트 AI들을 구현해서 턴수를 줄여 보자!!**

BattleShip Game Goal!

〈 목표 〉

1. C++의 객체 지향을 활용하여 프로그래밍 한다. (객체, 상속, 다형성, 캡슐화 등)
2. 확장이 용이 하도록 프로그래밍 한다. (ex. 쿠루저를 한대 더 추가한다면?)
3. 평균 턴 수를 최대한 줄이는 알고리즘을 제작한다.

첨부: 첨부한 샘플 프로그램의 결과를 참조하라.

Are you Ready?

Project #1
구현 및 제출

Project #1 : 배틀쉽 화면 구성 및 수동 플레이 구현

본 과제에서는 텍스트 사용자 인터페이스(Text User Interface, TUI) 라이브러리 중 하나인 ncurses 라이브러리를 이용하여 배틀쉽 게임의 화면을 구성하고, 수동 플레이 게임 로직을 구현 한다.

1. 과제 소개

본 과제에서는 텍스트 사용자 인터페이스(Text User Interface, TUI) 라이브러리 중 하나인 ncurses를 이용하여, 배틀쉽 게임 화면을 Fig1과 같이 구성하고, 게임을 수동으로 플레이하는 단계까지 로직과 자료구조 및 알고리즘을 구현한다.

2. 과제 목표

본 과제의 목표는 중규모 이상의 프로그램을 객체지향언어인 C++로 개발함에 있어 필요한 개발 요소를 학습하는 것이다.

- 요구 사항을 만족하는 프로그램 개발 : 복잡한 요구 사항을 분석하여 이를 만족하는 프로그램을 개발할 수 있는 능력을 확보한다.
- 외부 라이브러리를 이용한 프로그램 개발 : ncurses와 같은 외부 라이브러리를 이용하여 프로그램을 개발할 수 있는 능력을 확보한다.
- 여러 파일을 이용한 C++ 프로그램 개발 : 하나의 소스파일로만 구성된 프로그램이 아니라, 클래스 별로 헤더파일과 소스 파일을 구성하여 일정 규모 이상의 프로그램을 개발할 수 있는 능력을 확보한다.
- Make를 이용한 C++ 프로그램 개발 : 여러 소스파일로 구성된 프로그램의 컴파일/링크를 편리하게 하기 위한 도구로서 make를 종종 이용한다. 간단한 Makefile 파일을 만들고 make를 활용하여 여러 소스파일로 구성된 C++ 프로그램을 개발하는 능력을 확보한다.

Project #1 : ncursese를 이용한 TUI 화면 구성

NCURSES의 설치

Ubuntu에서 ncurses를 설치하기 위해서는 다음의 명령을 터미널에서 수행하도록 한다.

```
sudo apt-get update
sudo apt-get install libncurses5-dev libncursesw5-dev
```

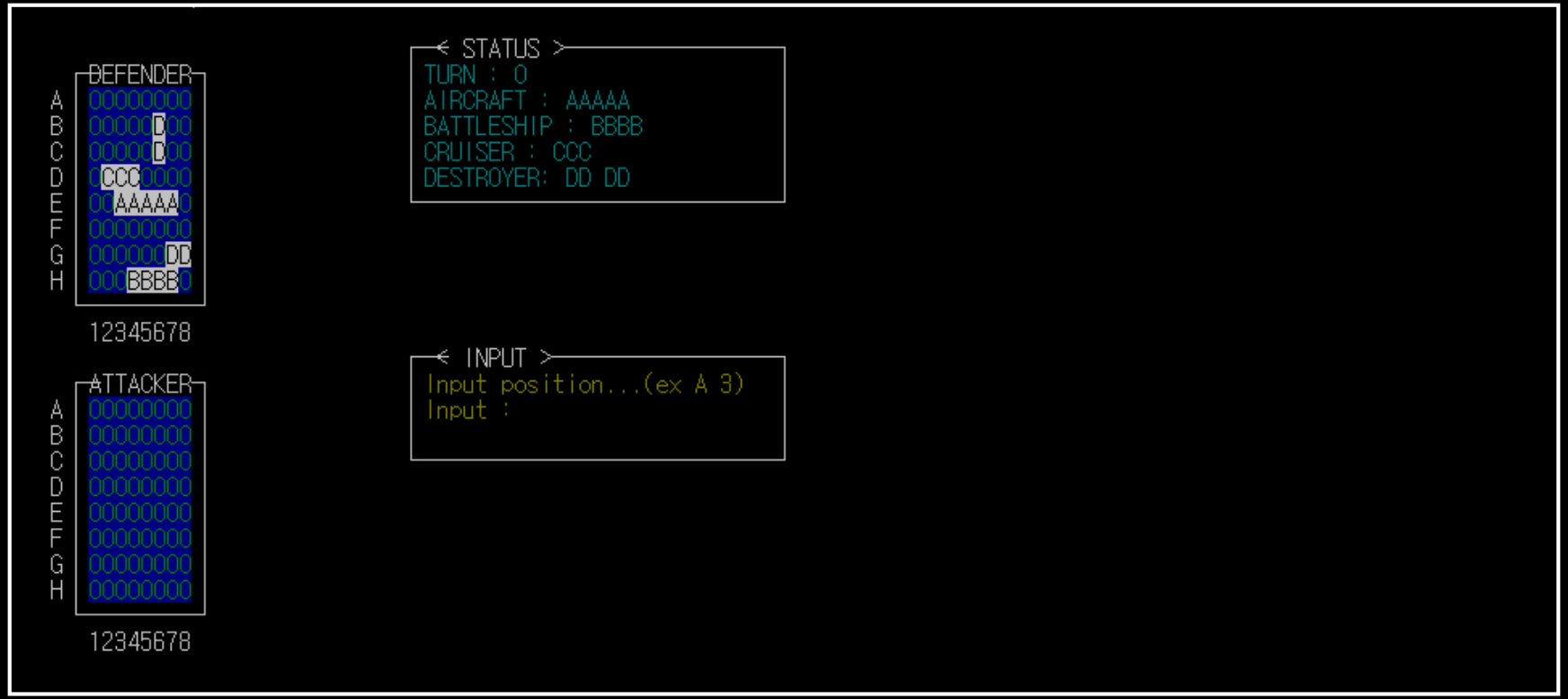
ncurses를 이용해서 작성한 소스코드 main.cpp가 있으면 다음과 같이 -lncurses 라이브러리 링크 옵션을 주고 컴파일 하도록 한다.

```
g++ main.cpp -lncurses
```

과제 요구 사항

다음의 세부 목표를 만족하는 프로그램을 디자인하도록 한다.

1. TUI 기반 테트리스 화면 구성 및 게임 로직 개발 (60점)
2. 다중 파일을 이용한 C++ 프로그램 개발 (20점)
3. Make를 이용한 C++ 프로그램 개발 (20점)



Project #1-1 : TUI 기반 배틀쉽 구현 (60점)

Project #1-1 : ncursese를 이용한 TUI 화면 구성

1. TUI 라이브러리 중 하나인 ncurses를 이용하여 게임화면을 Fig.1과 같이 구성한다. 배틀쉽 게임의 보드판의 규격인 8x8만을 준수하고, 그 외에는 자신의 개성에 맞추어서 구성해도 좋다.

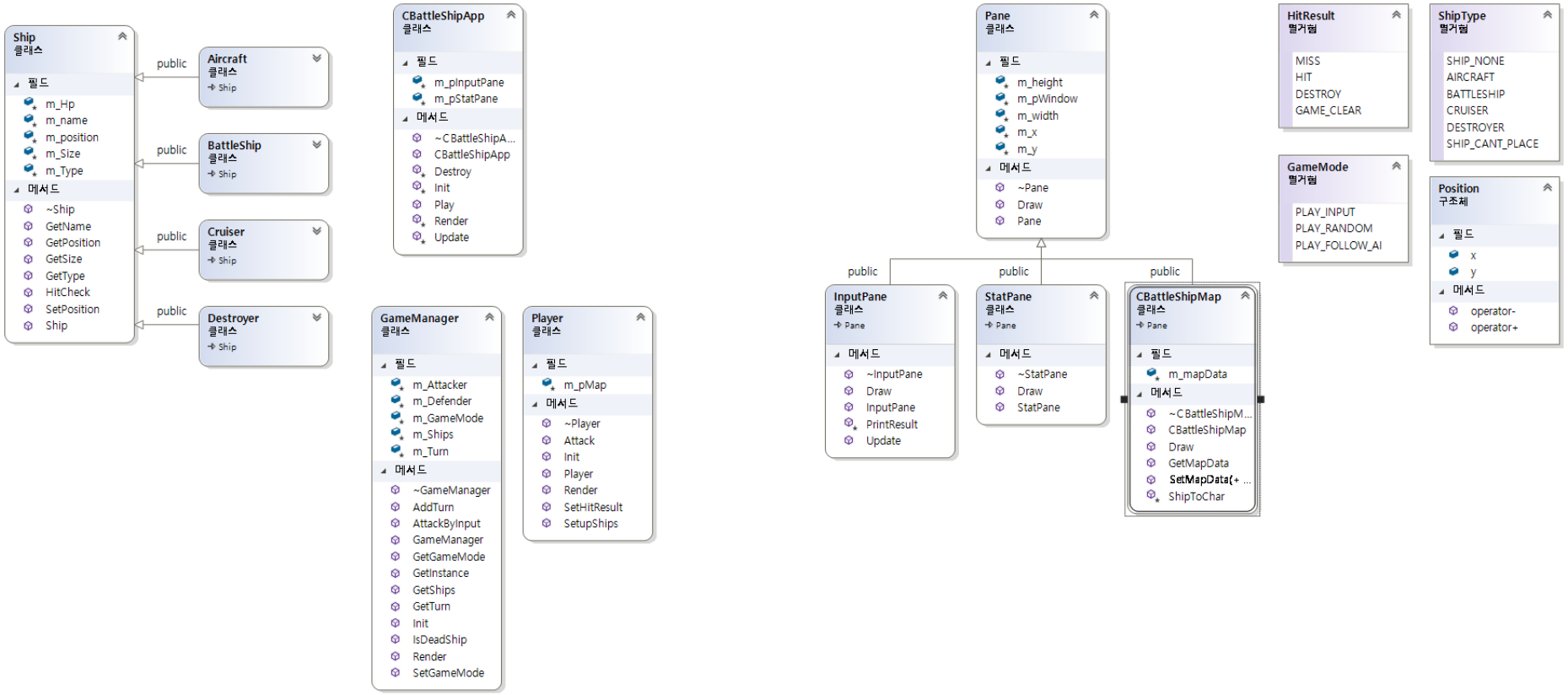
- TUI 모드 사용 시작 /종료
  프로그램 시작 시에 TUI mode를 시작하고 프로그램 종료 시에 TUI mode를 종료하도록 한다.
- 컬러쌍(color pair) 설정 기능
  Fig.1과 같이 각 윈도우의 화면 구성에 사용할 컬러 쌍들을 설정한다.
- 윈도우들의 생성 기능
  - 배틀쉽 게임 맵 화면을 구성하는 MapPane 생성
    - 배틀쉽 게임의 Map을 표시하는 윈도우이다.
    - 8X8의 칸으로 맵을 구성한다. 기본 값은 'O'로 채워 넣는다.

  - 배들의 격추 상태를 표시하는 StatusPane 생성
    - 각 배들의 상태를 표시하는 윈도우이다.
    - 본 과제에서는 Pane만 생성하고 내용을 텍스트로 출력만 한다.

  - 유저의 입력을 받을 InputPane 생성
    - 유저의 좌표 입력을 받을 윈도우이다.
    - 본 과제에서는 Pane만 생성하고 내용을 텍스트로 출력만 한다.

Project #1-1 : 배틀쉽 게임 로직 구현

1. 프로젝트 구성은 첨부한 UML 구조를 참고 한다. 다만 똑같이 제작할 필요는 없고, 가이드로 삼아서 아래 로직을 개별적으로 구현하면 된다.
2. GameManager 클래스가 존재한다. 이 클래스는 실제 배치된 배의 정보들 및 소요 턴수등의 게임의 전체적인 정보와 게임 로직을 관장한다.
3. 두명의 Player 객체를 생성한다. 하나는 Defender, 다른 하나는 Attacker가 된다.
4. Defender
    - Defender는 처음 시작할 때 맵에 자동으로 GameManager가 가진 배들을 랜덤 배치하는 역할을 담당한다.
    - 배치할 배는 Aircraft(5칸), BattleShip(4칸), Cruiser(3칸), Destroyer(2칸) X 2개의 총 5대를 배치한다. (Destroyer만 2대)
    - 배들의 위치와 방향(가로/세로)은 랜덤으로 배치 하며, 겹치거나 일부분이 맵밖으로 나가면 안된다.
5. Attacker
    - Attacker는 공격하는 객체이다. 매턴 공격 좌표를 결정하여 공격하고, 공격한 결과를 맵에 표시 체크 한다.
    - 1단계 프로젝트 : 수동으로 공격 좌표를 입력받기 때문에 Attacker가 하는일은 공격 결과를 맵에 표시하는것 뿐이다.
    - 2단계 프로젝트 : AI로 다음 공격 좌표를 계산하는 로직이 추가 되어야 한다.
6. Player 클래스는 각자의 Map 객체를 가지고 있다. Map 객체는 8X8로 구성된 2차원 배열을 가지고 있다.
7. 매 턴 유저로 부터 공격 좌표(ex. A3, b5…)를 유저에게 키보드로 입력 받아서, 공격 결과를 Attacker의 맵에 표시한다.
    - 공격 결과(HitResult)는 MISS / HIT / DESTROY의 세가지가 있다.
    - DESTROY의 경우에는 어떤 배가 격추되었는지 정보를 반환해야 한다.
8. 모든 배가 격추 될 때까지 턴을 반복하고, 모든 배가 격추되면 게임을 종료 한다.


Ship
클래스
필드
m_Hp
m_name
m_position
m_Size
m_Type
메서드
~Ship
GetName
GetPosition
GetSize
GetType
HitCheck
SetPosition
Ship
public
Aircraft
클래스
Ship
public
BattleShip
클래스
Ship
public
Cruiser
클래스
Ship
public
Destroyer
클래스
Ship
CBattleShipApp
클래스
필드
m_pInputPane
m_pStatPane
메서드
~CBattleShipA...
CBattleShipApp
Destroy
Init
Play
Render
Update
GameManager
클래스
필드
m_Attacker
m_Defender
m_GameMode
m_Ships
m_Turn
메서드
~GameManager
AddTurn
AttackByInput
GameManager
GetGameMode
GetInstance
GetShips
GetTurn
Init
IsDeadShip
Render
SetGameMode
Player
클래스
필드
m_pMap
메서드
~Player
Attack
Init
Player
Render
SetHitResult
SetupShips
Pane
클래스
필드
m_height
m_pWindow
m_width
m_x
m_y
메서드
~Pane
Draw
Pane
public
InputPane
클래스
Pane
메서드
~InputPane
Draw
InputPane
PrintResult
Update
public
StatPane
클래스
Pane
메서드
~StatPane
Draw
StatPane
public
CBattleShipMap
클래스
Pane
필드
m_mapData
메서드
~CBattleShipM...
CBattleShipMap
Draw
GetMapData
SetMapData(+ ...
ShipToChar
HitResult
멀거형
MISS
HIT
DESTROY
GAME_CLEAR
GameMode
멀거형
PLAY_INPUT
PLAY_RANDOM
PLAY_FOLLOW_AI
ShipType
멀거형
SHIP_NONE
AIRCRAFT
BATTLESHIP
CRUISER
DESTROYER
SHIP_CANT_PLACE
Position
구조체
필드
x
y
메서드
operator-
operator+

Project #1-2 :다중 파일을 이용한 C++ 프로그램 개발 (20점)

1. 하나의 소스코드(예:main.cpp)로 프로그램을 개발하게 되면 협업을 통한 프로그램 개발이 어려워지고, 관리가 복잡해진다는 단점이 있다. 간단한 프로그램을 개발하는 경우라도 각 클래스별로 파일을 분할해서 개발하는 경우가 일반적이다.
   C++ 언어를 이용한 개발에서는 하나의 클래스는 헤더 파일 하나와 소스 파일 하나로 구성하는 것이 일반적이다. 예를 들어, Pane 클래스는 선언은 Pane.h라는 헤더 파일에, 구현은 Pane.cpp 소스파일에 표현되도록 한다.
   본 과제에서는 등장하는 모든 클래스에 대해 각 클래스가 자신만의 헤더파일과 소스파일을 가지도록, 다음과 같이 다중 파일을 이용하여 C++ 프로그램을 개발하도록 한다.
   다중 파일로 개발할 경우, 헤더파일이 중복 포함될 수 있기 때문에 헤더 파일이 중복 포함되지 않도록 모든 헤더 파일에 다음과 같이 #ifndef #define #endif 문을 적절하게 활용하도록 한다. 자세한 사항은 교재 323-324 페이지를 참고 한다. (대신 #pragma once 를 사용해도 된다)

```
// Pane.h
#ifndef __PANE_H__
#define __PANE_H__

...

#endif //__PANE_H__
```

Project #1-3 :Make를 이용한 C++ 프로그램 개발 (20점)

프로그램 개발 규모가 어느 정도 커지면 앞서 설명만 바와 같이 다중 파일로 개발하는 경우가 많아지게 된다. 이런 경우 매번 다음과 같이 모든 소스파일들을 g++ 컴파일러에게 알려줘서 프로그램을 컴파일해야 하는데, 이는 귀찮을 뿐만 아니라, 실수를 유발하기 쉽다.

```
g++ -o BattleShip BattleShip.cpp Pain.cpp MapPain.cpp StatusPain.cpp ... InputPain.cpp - lncurses
```

이러한 작업을 간단하게 해주는 도구로 널리 쓰이는 것이 make이다. Make는 현재 디렉토리에 Makefile의 내용을 기반으로 **작업을** 수행하는 도구인데 일반적으로 C++ 프로그램 개발에서는 다음과 같이 활용하도록 한다.

프로그램 소스 코드가 있는 디렉토리에 Makefile이라는 텍스트 파일을 만든다.

Makefile의 내용을 g++ 컴파일 내용으로 채운다

매번 새롭게 컴파일이 필요한 경우 터미널에서 간단한게 make를 수행하면 Makefile의 내용에 따라 컴파일이 된다.

Make를 활용하기 위해서는 Makefile을 잘 작성해야 하는데 다음은 C++ 프로그램 개발에서 널리 쓰이는 일반적인 형태의 Makefile이다.

```
# Makefile example
CC = g++
TARGET = tetris
SOURCES = tetris.cpp \
          Pane.cpp \
          BoardPane.cpp \
          InfoPane.cpp \
          HelpPane.cpp \
          StatPane.cpp \
          NextPane.cpp
LDFLAGS = -lncurses

all:
  $(CC) -o $(TARGET) $(SOURCES) $(LDFLAGS)

clean:
  rm -rf *.o $(TARGET)
```

본 과제에서는 위와 같이 자신이 작성한 다중 파일들로 구성된 프로그램이 make를 통해 정상적으로 컴파일 되도록 Makefile을 작성하는 것을 목표로 한다. Ubuntu 터미널 환경에서 make로 정상적으로 컴파일 되도록 하는 것이 목표이다.

Project #1 : 참고할 정보

- 과제와 함께 첨부된 ncurses library 튜터리얼 자료를 참고한다.
  - 배틀쉽 프로젝트 기본 UI 구성에 관한 참고자료 샘플을 PT에 첨부한다.
- Ncurses에 대해 더 자세히 알고 싶은 경우, 아래 사이트를 참고한다.
  - http://pubs.opengroup.org/onlinepubs/7908799/xcurses/curses.h.html
  - (한국어) NCURSES-Programming-HOWTO
- Make에 대해 저 자세히 알고 싶은 경우, 아래 사이트를 참고한다.
  - 위키백과 make
  - GNU Make 강좌 – 임대영

Project #1 :과제 제출 방법 (매우 중요!!!!)

- 프로그램의 각 클래스와 함수들은 첨부한 UML을 참고로 하여 구성하도록 한다. 단 반드시 UML과 같은 필요는 없다. 다만 각 기능별로 클래스와 함수들을 구별하도록 한다.
- **우측과 같은 형식으로 각 헤더파일에 자신의 학번과 이름을 기록한다.**
- 프로젝트는 제출 기간까지 가상 대학에 제출 하도록 한다.
- 과제 코드는 **Ubuntu 환경에서 make 명령으로 컴파일 가능**하도록 작성한다.
- 과제 코드는 다음의 파일들은 하나의 압축파일로 묶어 tar.gz 파일 형식이나 표준 zip 파일 형식으로만 제출하도록 한다. 이때, 압축 파일의 이름은 반드시 "OOOOOOO_PROJ_01.tar.gz (OOOOO은 자신의 학번)"과 같이 자신의 학번이 드러나도록 제출한다.
  1) 모든 소스파일 (*.h, *.cpp)
  2) Makefile
- 과제에 관한 질문은 오피스아워를 활용하도록 한다. 교육조교(teaching assistant, TA)에 메일로 약속 시간을 정한 후, 교육 조교가 있는 연구실로 방문하여 물어보는 것도 매우 권장하는 방법이다.

```
1  // C++ BattleShip 프로젝트
2  // 작성 일자 : 2018-05-15
3  // 학번 : 2017490 이름 : 박민근
4
5  #pragma once
6  #include <vector>
7  #include "Player.h"
8  class Ship;
9
10 class GameManager
11 {
12 public:
13     GameManager();
14     ~GameManager();
15
16     static GameManager* GetInstance()
17     {
18         static GameManager instance;
19         return &instance;
20     }
21
22     void Init();
23     void Render();
24
25     HitResult AttackByInput(const Position& pos, s
```

Project #2
AI 기능 구현하기

Project #2 : 배틀쉽 AI 플레이 구현하기

1단계에서 완성한 배틀쉽 프로젝트를 수동 플레이가 아닌, AI가 진행하는 자동 플레이 버전으로 수정 / 구현 한다.

1. **과제 소개**
   1단계 과제에서 수동으로 플레이하는 배틀쉽 프로젝트를 완성하였다. 기존의 프로젝트에서 Attaker의 공격 알고리즘을 결정하는 AI를 구현하여, 유저의 입력이 없이 자동으로 진행되는 배틀쉽 게임 프로젝트를 제작하여 제출 한다.

2. **과제 목표**
   간단한 AI 로직을 구현하고, 프로젝트에 적용할 수 있다. 이 과정에서 객체 지향 프로그래밍, 자료구조, 알고리즘등을 활용하는 능력을 확보할 수 있다.

   - 간단한 AI 로직을 구현하는 프로그래밍 능력을 학습한다.
   - 기존 프로젝트 코드를 요구 사항에 따라서 수정/변경할 수 있는 경험과 능력을 학습한다.
   - 효율 적인 알고리즘 구현에 대한 고민과 자료구조의 활용 방법을 학습한다.

Project #2-1 : 랜덤으로 공격 좌표 결정하기

기본적인 구성으로 공격 위치를 랜덤으로 결정하여, 공격 한다.

1. **사용자의 입력을 배제 한다**
   기본 프로젝트에서 유저의 입력을 받았던 부분을 수정하여, Attacker가 매 턴 자동으로 공격 위치를 결정해서 게임을 진행하도록 수정 한다.
2. **공격 알고리즘**
   - Attacker가 매 턴 공격 위치를 랜덤으로 결정한다.
   - 이미 공격한 위치는 공격 위치에서 제외한다.
   - 모든 배를 격추할 때까지 공격을 반복 한다.
3. **게임 10판을 반복 수행하고, 최종적으로 평균 턴 수를 출력한다.**
   - 매 턴 위치를 결정하여, 자동 공격한 결과를 Attacker의 Map에 표시한다.
   - 게임이 자동으로 수행 되기 때문에, 순식간에 게임이 자동으로 진행되고, 결과를 계산할 수 있다.
   - 매판마다 게임의 종료 될 때까지의 턴 수를 계산하여, 10판을 반복 수행한후에 평균 턴 수를 화면에 출력 한다.

Project #2-2 : 개선된 AI - 히트 이후 효율 적인 공격 위치 결정 하기

랜덤으로 공격 위치를 결정하는 것은 아주 단순한 AI이다. 이 AI를 개선하여, 좀더 인간이 발상할 수 있는 효율적인 추적 알고리즘을 구현 한다.

1. 과제 목표

기본적으로는 공격 위치를 랜덤으로 공격 한다. 그리고 Hit 한 다음 부터는 아래 알고리즘으로 다음 공격 위치를 결정 한다.

- Hit가 되었다면, 배의 크기가 2칸 이상이기 때문에, 인접한 칸에 배의 다른 부분이 존재 한다.
- 4방향의 인접한 칸을 하나 골라서 다음 공격 위치로 결정한다.
- MISS가 발생하였다면, 다른 방향으로 공격 위치를 결정한다.
- 연속된 위치가 히트 하였다면, 그 방향으로 배가 있을 가능성이 높다. 그 방향을 따라서 (Follow) 다음 공격 위치를 결정 한다.
- 진행 방향에 따라서, 다음 공격 위치를 결정하여 공격 한다. MISS가 나오거나 DESTROY가 나올 때까지 반복 한다.
- 파괴되지 않았는데 MISS가 발생하였다면, 시작점에서 진행 방향의 반대 방향으로 공격을 진행한다.
- 반대 방향으로 계속 갔는데도, DESTROY가 아니라 MISS가 나온다면, 2개 이상의 배가 연속으로 배치되어 있는 것이다. 이 경우에 대해서도 처리 한다.

Project #2-3 : 공격할 위치에 대해서 가능성 맵을 만들어서 적용하기

추적 알고리즘은 HIT한 이후에 효율 적이다. HIT 이전에 공격할 위치를 결정할 때, 맵에 배가 배치되어있을 가능성을 판단하여 공격하는 스마트 한 AI를 구현 한다.

1. **과제 소개**
   만약 맵에 첫 턴에 공격을 할 때, 맵의 맨 구석보다는 중앙쪽이 배가 배치되어 있을 가능성이 높다고 판단을 할 것이다. 이와 같은 판단을 기반으로 한 <배가 배치 되어 있을 가능성>에 대한 확률을 계산하여 가능성이 가장 높은 위치 부터 공격하는 효율 적인 알고리즘을 구현 한다.

2. **과제 목표**
   HIT가 발생한 이후에는 이전의 추적 알고리즘으로 효율적으로 구현할 수 있다. 하지만 HIT 이전에 공격 위치를 랜덤으로 결정하는 것은 효율적이지 못하다. 인간이 기본적으로 떠올리는 확률적인 부분을 구현하여, 공격 위치에 적용하는 AI를 구현 한다.

   - 맵의 각 영역에 대해서 확률을 계산하는 확률맵(가능성 맵)을 만든다.
   - 맵의 각 위치는 배가 놓여 있을 가능성이 각각 다르다. 예를 들어 배의 크기를 고려하면, 맨 구석에 있을 확률보다는 정중앙의 위치가 배가 있을 확률이 높다.
   - 공격한 위치에 따라서, 맵의 각 영역의 확률은 매 턴 변화 한다. 예를 들어 공격한 위치 A, B가 있다고 할 때 두 위치의 간격이 3칸이라고 한다면, 가운데에 배치 되어 있을 가능성이 있는 배는 2칸짜리인 DESTROYER 뿐이다. 만약 반대 위치에 6칸이 비어 있다면, 그 쪽에 배가 배치되어 있을 확률이 높은 것이다.
   - 위와 같은 알고리즘을 기반으로 하여, 맵의 모든 영역에 대해서 배가 배치될 수 있는 가능성을 중첩하여 계산하면, 맵의 각 영역에 배가 배치될 수 있는 확률을 계산할 수 있다.
   - 계산한 확률을 기반으로 하여 다음 공격 위치를 결정한다.

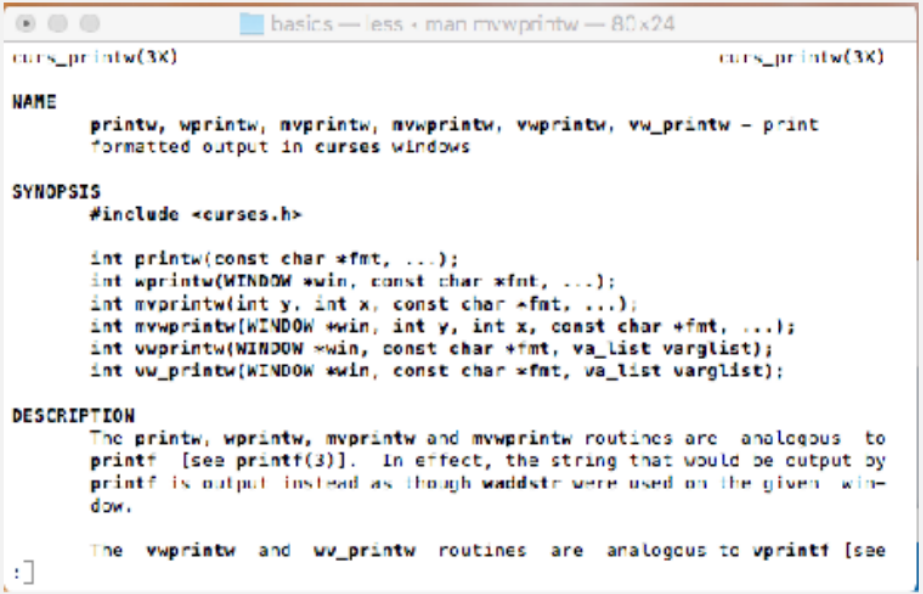
Project #2 : 평가 방법 및 제출 방법

- **프로젝트 #2의 제출 방법은 Project #1의 제출 방법과 동일 하다.**
- 매 턴 자동 공격 하는 과정을 화면에 출력 한다.
- **기본적으로는 2-1 랜덤 공격하여 자동으로 진행되는 프로젝트까지 만드는 것을 프로젝트의 기본 목표로 한다.**
- 2-2 추적 알고리즘을 구현하면 **추가 점수**를 부여 한다.
- 2-3 가능성 맵 알고리즘까지 구현하면 **추가 점수**를 부여 한다.
- 효율적인 알고리즘이라는 것을 평가하는 방법은 최종적으로 계산되는 평균 턴 수로 판단할 수 있다.

NCURSES

ncurses

- text-based user-interface library
  - GPU ncurses http://www.gnu.org/software/ncurses/
  - NCURSES Programming HOWTO http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/ : with helpful examples
  - 한국어 번역 https://wiki.kldp.org/wiki.php/NCURSES-Programming-HOWTO
  - $man ncurses on linux command prompt shows manual page
  - $man mvvline etc.



Hello_world.cpp for ncurses

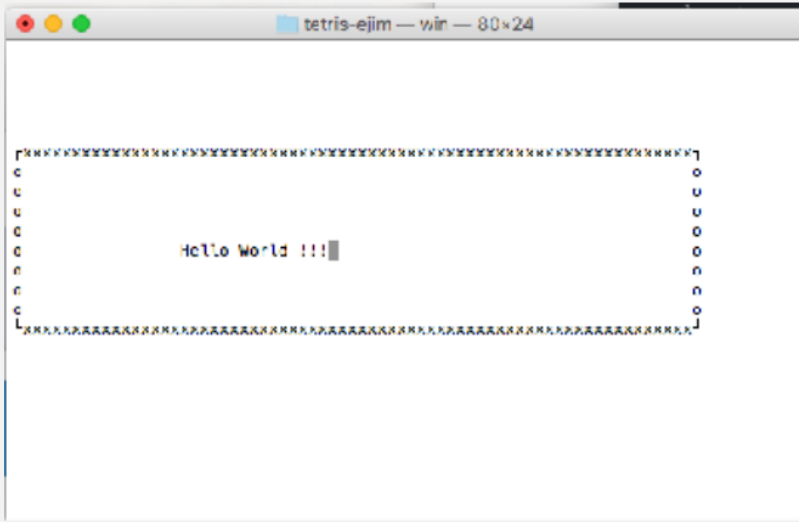
- compile command
  - g++ -o hello hello.cpp -lncurses

```
#include <ncurses.h>
int main()
{
        initscr();              /* Start curses mode            */
        printw("Hello World !!!");      /* Print Hello World            */
        refresh();              /* Print it on to the real screen */
        getch();                /* Wait for user input */
        endwin();               /* End curses mode              */

        return 0;
}
```

creating windows using ncurses library

- window is a non-overlapping part of text screen.
- when curses is initialized, a default window named stdscr which represents the whole size of window in which your terminal (xterm) is running.

```
#include <ncurses.h>
int main()
{
        initscr();
        refresh(); // 꼭 필요함!!
        int height = 10, width = 70, x = 0, y = 5;
        WINDOW *w = newwin(height, width, y, x);
         box(w, 'o', 'x');
        mvwprintw(w, height/2, width/4, "Hello World !!!");
        wrefresh(w);

        getch();                    /* Wait for user input */
         delwin(w);
        endwin();                           /* End curses mode

        return 0;
}
```

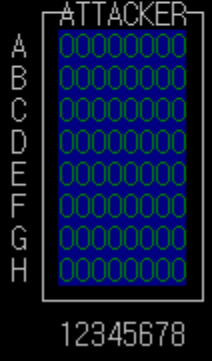


1. 기본 UI 구성

<< Battle Ship Game >>

< STATUS >
TURN : 0
AIRCRAFT : AAAAA
BATTLESHIP : BBBB
CRUISER : CCC
DESTROYER: DD DD

< INPUT >
Input position...(ex A 3)
Input :



Main.cpp

```
1   // BattleShip.cpp: 콘솔 응용 프로그램의 진입점을 정의합니다.
2   //
3
4   #include "stdafx.h"
5   #include "CBattleShipApp.h"
6
7   int main()
8   {
9       CBattleShipApp battleShip;
10      battleShip.Play();
11      return 0;
12  }
```

BattleShipApp.h

```
class CBattleShipApp
{
public:
    CBattleShipApp();
    ~CBattleShipApp();

    void Play();

protected:
    void Init();
    void Render();
    void Destroy();

protected:
    CBattleShipMap* m_pMap;
    StatPane* m_pStatPane;
    InputPane* m_pInputPane;
};
```

BattleShipApp.cpp

```cpp
void CBattleShipApp::Init()
{
    initscr();
    start_color();
    cbreak();
    refresh();

    // 컬러 세팅
    init_pair(1, COLOR_GREEN, COLOR_BLACK);
    init_pair(2, COLOR_CYAN, COLOR_BLACK);
    init_pair(3, COLOR_YELLOW, COLOR_BLACK);

    m_pMap = new CBattleShipMap();
    m_pStatPane = new StatPane(30, 3, 30, 6);
    m_pInputPane = new InputPane(30, 15, 30, 4);
}
```

```cpp
void CBattleShipApp::Play()
{
    Init();
    Render();
    Destroy();
}

void CBattleShipApp::Render()
{
    mvprintw(1, 1, "<< Battle Ship Game >>");

    m_pMap->Draw();
    m_pStatPane->Draw();
    m_pInputPane->Draw();

    refresh();
}

void CBattleShipApp::Destroy()
{
    getch();
    endwin();
    delete m_pMap;
}
```

Pane.h

```h
// 화면을 구성하는 Pane의 부모 클래스
class Pane
{
public:
    Pane(int x, int y, int widht, int h
    virtual ~Pane();

    virtual void Draw();

protected:
    int m_width, m_height;
    int m_x, m_y;
    WINDOW* m_pWindow;
};
```

```cpp
Pane::Pane(int x, int y, int width, int height)
    :m_x(x), m_y(y), m_width(width), m_height(heigh
{
    m_pWindow = newwin(height, width, y, x);
    box(m_pWindow, 0, 0);
    wrefresh(m_pWindow);
}

Pane::~Pane()
{
    delwin(m_pWindow);
}

void Pane::Draw()
{
    box(m_pWindow, 0, 0);
    wrefresh(m_pWindow);
}
```

Pane.cpp

```
Pane::Pane(int x, int y, int width, int height)
    :m_x(x), m_y(y), m_width(width), m_height(height)
{
    m_pWindow = newwin(height, width, y, x);
    box(m_pWindow, 0, 0);
    wrefresh(m_pWindow);
}

Pane::~Pane()
{
    delwin(m_pWindow);
}

void Pane::Draw()
{
    box(m_pWindow, 0, 0);
    wrefresh(m_pWindow);
}
```

BattleShipMap.h

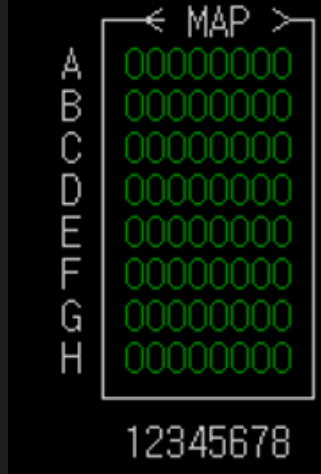
```
#include "Pane.h"

#define MAP_SIZE 8

// 게임 화면 맵을 표시하는 클래스
class CBattleShipMap : Pane
{
public:
    CBattleShipMap();
    ~CBattleShipMap();

    void Draw();

protected:
    char m_mapData[MAP_SIZE][MAP_SIZE];
};
```

BattleShipMap.cpp #1

```cpp
CBattleShipMap::CBattleShipMap()
    :Pane(4, 4, MAP_SIZE + 3, MAP_SIZE + 2)
{
    for (int i = 0; i < MAP_SIZE; i++)
    {
        for (int j = 0; j < MAP_SIZE; ++j)
        {
            // 맵 데이터 초기화
            m_mapData[i][j] = '0';
        }
    }

    // 칸 구별 이름
    for (int i = 0; i < MAP_SIZE; ++i)
    {
        mvprintw(i + 1 + m_y, m_x - 1, "%c", 'A' + i);
        mvprintw(m_y + m_height, m_x + 2 + i, "%d", 1 + i);
    }

    // 타이틀
    mvwprintw(m_pWindow, 0, 3, "< MAP >");
}
```

BattleShipMap.cpp #2

```cpp
CBattleShipMap::~CBattleShipMap()
{
}

void CBattleShipMap::Draw()
{
    wattron(m_pWindow, COLOR_PAIR(1));
    for (int i = 0; i < MAP_SIZE; ++i)
    {
        for (int j = 0; j < MAP_SIZE; ++j)
        {
            mvwprintw(m_pWindow, i + 1 , j + 2 , "%c", m_mapData[i][j]);
        }
    }
    wattroff(m_pWindow, COLOR_PAIR(1));

    wrefresh(m_pWindow);
}
```



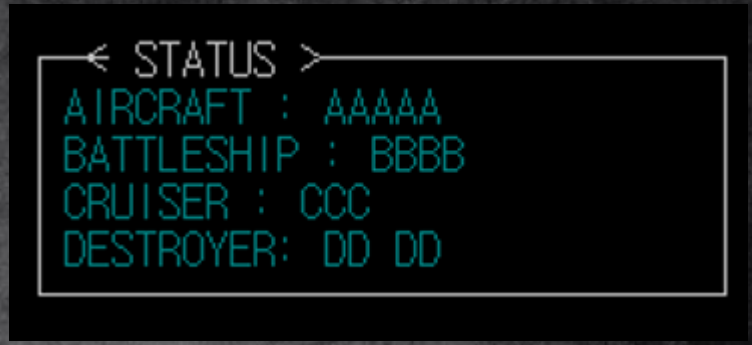
StatPane.h

```h
#include "Pane.h"

// 스태터스를 표시하는 윈도우
class StatPane : Pane
{
public:
    StatPane(int x, int y, int width, int height);
    ~StatPane();

    virtual void Draw();
};
```

```cpp
StatPane::StatPane(int x, int y, int width, int height)
    :Pane(x, y, width, height)
{
    // 타이틀
    mvwprintw(m_pWindow, 0, 3, "< STATUS >");
}
```

StatPane.cpp

```
void StatPane::Draw()
{
    wattron(m_pWindow, COLOR_PAIR(2));
    mvwprintw(m_pWindow, 1, 2, "AIRCRAFT : AAAAA");
    mvwprintw(m_pWindow, 2, 2, "BATTLESHIP : BBBB");
    mvwprintw(m_pWindow, 3, 2, "CRUISER : CCC");
    mvwprintw(m_pWindow, 4, 2, "DESTROYER: DD DD");
    wattroff(m_pWindow, COLOR_PAIR(2));

    wrefresh(m_pWindow);
}
```



StatPane.h

```
#include "Pane.h"
class InputPane :
    public Pane
{
public:
    InputPane(int x, int y, int width, int height);
    ~InputPane();

    virtual void Draw();
};
```

StatPane.cpp

```cpp
InputPane::InputPane(int x, int y, int width, int height)
    :Pane(x, y, width, height)
{
    // 타이틀
    mvwprintw(m_pWindow, 0, 3, "< INPUT >");
}


InputPane::~InputPane()
{
}

void InputPane::Draw()
{
    wattron(m_pWindow, COLOR_PAIR(3));
    mvwprintw(m_pWindow, 1, 2, "Input position...(ex A 3)");
    mvwprintw(m_pWindow, 2, 2, "Input : ");
    wattroff(m_pWindow, COLOR_PAIR(3));

    wrefresh(m_pWindow);
}
```