



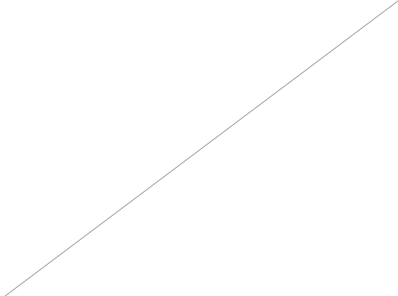
HeadFirst_Invader



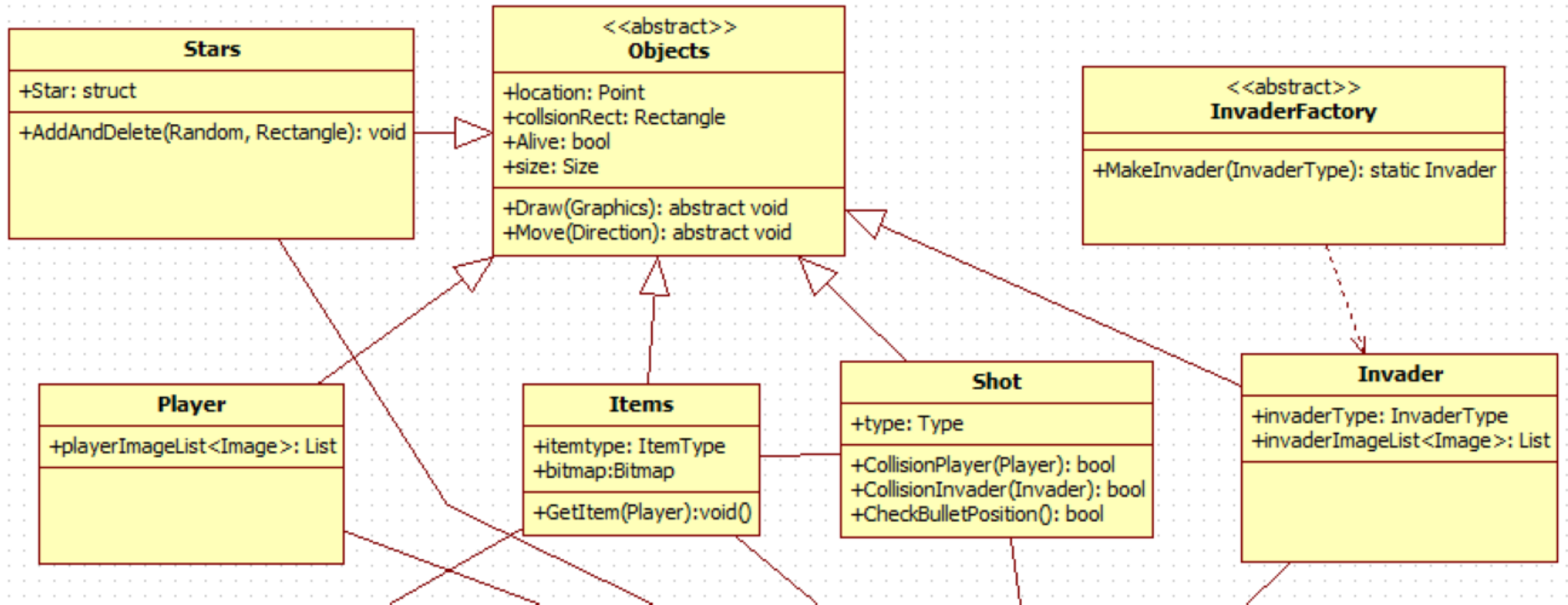
이재건



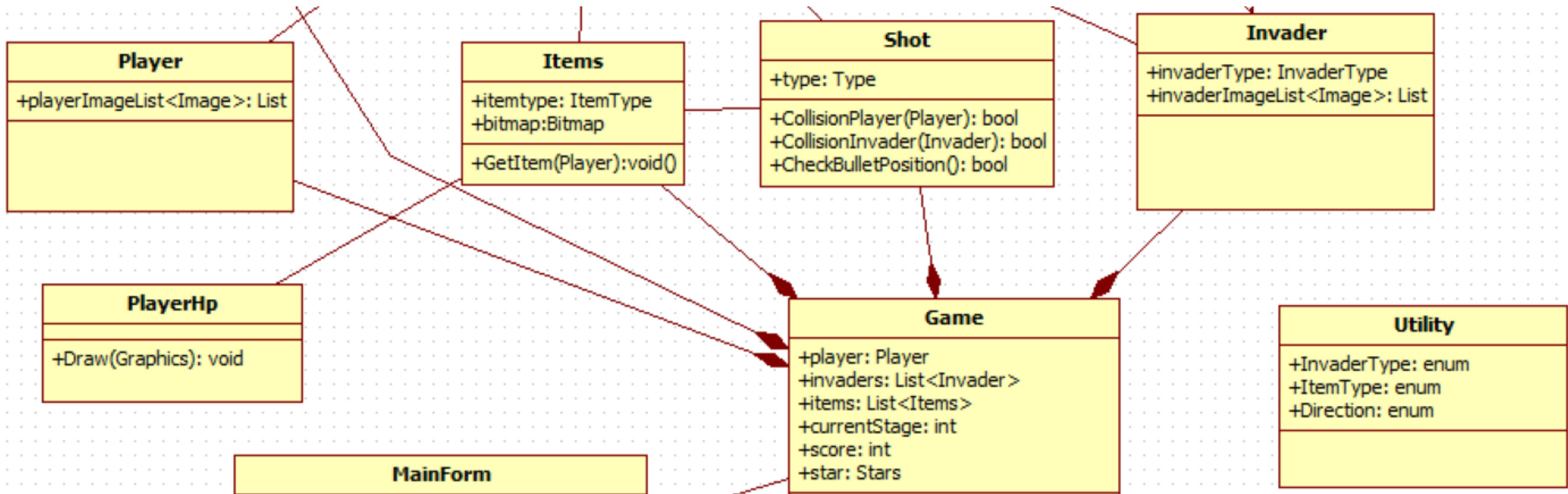
Contents

- 클래스 다이어그램
 - 추가 구현
 - 핵심 코드
 - 실습**#3**을 마치며
- 

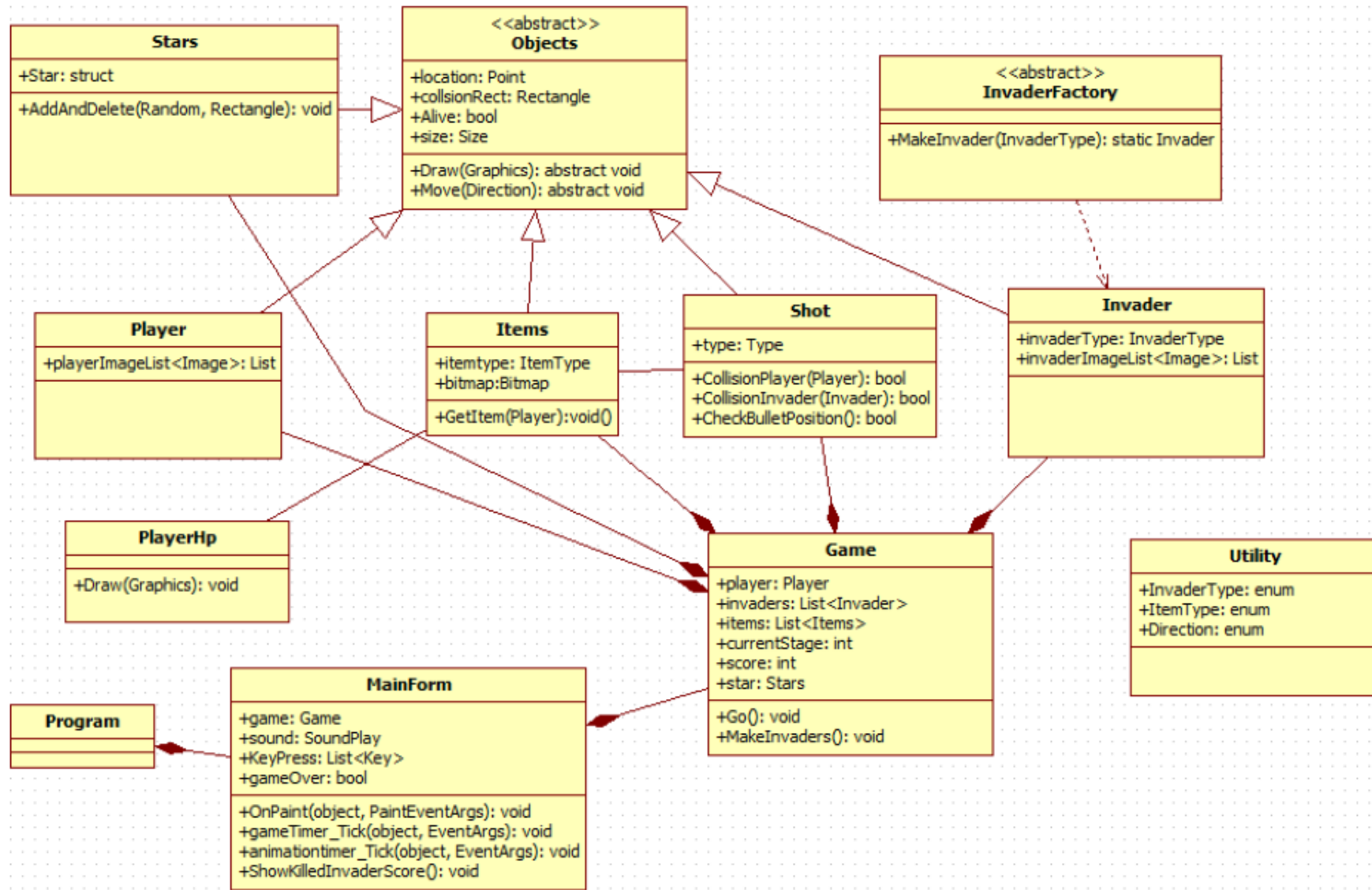
클래스 다이어그램_1



클래스 다이어그램_2



클래스 다이어그램_3



추가 구현

- 스테이지가 나뉘어져 있어서 스테이지가 증가함에 따라 몬스터의 속도가 증가
- -> Animation_timer의 interval 을 현재 stage의 값*10 만큼 차감
- 몬스터 제거 시 일정 확률로 아이템 드랍
- ->아이템은 체력증가와 총알 갯수 증가 2가지 종류
- 피격시 화면 흔들림 효과 , 타격시 해당 Invader의 점수를 2초간 보여줌
- ->가장 하단의 invader는 10점으로 시작하여 위로 올라갈수록 +10씩 증가

- 배경 사운드 추가
- ->C#에서 제공하는 SoundPlayer 클래스 이용하여 back ground music 추가
- 명예의 전당 시스템
- ->랭킹 시스템으로 게임 시작 전 확인 할 수 있으며 MessageBox로 출력하여 정보를 제공
- ->점수에 대한 정보는 text파일로 프로젝트 파일에 저장

핵심 코드_1

- 인베이더 팩토리

팩토리 메서드 패턴을 이용하여 enum 값에 따라 분류하여 invader 종류를 나누었습니다.

- 충돌처리

timer를 이용하여 game.Go()함수를 호출하면 아래와 같이 4개의 메서드가 실행됩니다. 충돌처리는 충돌하는 2개의 객체의 각각의 Location값과 rectangle 사이즈를 이용하여 처리합니다.

```
public void Go()
{
    PlayerAndInvaderCollisionCheck();

    PlayerBulletCollisionCheck();

    InvaderBulletCollisionCheck();

    PlayerItemCollisionCheck();
}
```

```
switch (invaderType)
{
    case InvaderType.Star:
        starxpos += invaderInterval;
        return new Invader(InvaderType.Star, starInvaderPos, StarInvaderScore);

    case InvaderType.Bug:
        bugxpos += invaderInterval;
        return new Invader(InvaderType.Bug, bugInvaderPos, BugInvaderScore);

    case InvaderType.Spaceship:
        spacexpos += invaderInterval;
        return new Invader(InvaderType.Spaceship, spaceInvaderPos, SpaceInvaderScore);

    case InvaderType.Satellite:
        satellitexpos += invaderInterval;
        return new Invader(InvaderType.Satellite, satelliteInvaderPos, SatelliteInvaderScore);

    case InvaderType.Saucer:
        saucerxpos += invaderInterval;
        return new Invader(InvaderType.Saucer, saucerInvaderPos, SaucerInvaderScore);

    default:
        return null;
}
```

핵심 코드_2

- 명예의 전당 시스템 구현코드

```
using (StreamReader textreader = new StreamReader(@"hallOfFame\hallOfFame.txt"))
{
    ...
    var sortingScore = from score in loadScore
                        orderby score descending
                        select score;
}
```

```
using (StreamWriter streamwriter = new StreamWriter(@"hallOfFame\hallOfFame.txt"))
{
    ...
}
```

Using 과 LINQ를 이용하여 text파일로 자료를 write 하고 text파일에 자료를 read하는 것을 통하여 데이터 저장
이때 LINQ를 이용해 파일 내부에 SCORE 값들을 내림차순으로 정렬하여 점수를 저장.

```
if (line[i] == '#')
{
    temp = line.Substring(i + 1, line.Length - (i + 1));
    loadScore.Add(Convert.ToInt32(temp));
}
```

If 문과 subString을 이용하여 점수만 가져오게 합니다.

핵심 코드_3

- 인베이더 타격 시 점수 시스템

->PlayerBullet이 invader와 충돌을
한다면 Game.killinvader의 bool값을
참으로 변경하면서 list에 해당
invader의 위치를 저장합니다.

이제 animation_timer에서 killinvader가
true라면 오른쪽 코드와 같이 label을
띄워주며 일정 tickcount 후 사라지게
합니다.

```
private void ShowKilledInvaderScore()
{
    //각가의 점수를 표기하는 기간이 다르므로 invader 클래스 자체가
    //이것에 대한 기능을 갖는것이 좋다고 생각!
    if (game.KillInvader)
    {
        for (int i = 0; i < game.InvaderScoreLabelList.Count; i++)
        {
            invaderscoreLabel[0].Location = game.InvaderScoreLabelList[i].Location;
            invaderscoreLabel[0].Text = game.InvaderScoreLabelList[i].Score.ToString();
            invaderscoreLabel[0].Visible = true;
            game.InvaderScoreLabelList.RemoveAt(i);
        }

        if (invaderscoreLabel[0].Visible)
        {
            invaderscoreLabel[0].Location = new Point(invaderscoreLabel[0].Location.X,
                invaderscoreLabel[0].Location.Y - 3);

            ShowScoreTick += 1;
            if (ShowScoreTick >= 10)
            {
                invaderscoreLabel[0].Location = new Point(-10, -10);
                invaderscoreLabel[0].Visible = false;
                game.KillInvader = false;
                ShowScoreTick = 0;
            }
        }
    }
}
```

실습#3을 마치며...

- 부족했던 부분:

- >자료구조(LINQ)를 이용하여 충돌처리를 하지 못했던 부분.
- >가급적 메소드 내부에서 직접 입력한 값[숫자로 값을 입력한 부분]을 최소한으로 사용하지 못한 부분.[충돌처리]
- >별을 이쁘게 못찍었던 부분.....
- >다른사람이 나의 코드를 보고 손쉽게 이해할 수 있는 가독성이 부족했던 부분.
- >Invader의 총알 발사 시 동시에 발사 될 경우가 존재하는데 이 문제를 고치지 못한 부분.

- 추가하고 싶은 부분:

- >보스몬스터 구현
- >피격&타격 시 사운드 추가
- >유도 미사일 구현



실습#3을 마치며...

Q&A

