졸업작품 최종보고서

-게임 개발 프로젝트-

“FuzZom” 개발 보고서



이름: 엄철용

학번: 92020329

학과: 게임소프트웨어학과

담당 지도 교수님: 박인규 교수님

제출일: 2024년 12월 6일

목록(1/2)

1. 프로젝트 배경
2. 프로젝트 목표
3. 프로젝트 설계
   1. 프로젝트 플로차트
   2. 프로젝트 '퍼지 논리' 알고리즘 설계
   3. 프로젝트 멀티플레이 설계
4. 프로젝트 구현
   1. 프로젝트 사용 기술
   2. 프로젝트 상세 기술
      1. 좀비 AI 시스템
      2. 프로젝트 맵 디자인
      3. 프로젝트 멀티플레이 기능 구현
      4. 프로젝트 아이템 기능 구현
      5. 프로젝트 UI 구현
      6. 프로젝트 플레이어 캐릭터 구현
      7. 프로젝트 플레이어 무기 구현

목록(2/2)

1. 최종 결과물
   1. 메인 화면
   2. 게임플레이 화면
   3. 게임 오버 화면
2. 문제점 및 개선방안
   1. 문제점
   2. 개선 방안
   3. 기타 피드백
3. 코드
   1. 퍼지논리
   2. Photon 관련 코드
   3. GameManager
   4. LivingEntity
   5. LobbyManager
   6. 좀비AI
   7. 좀비AI 스폰
4. 참고문헌
5. 프로젝트 배경

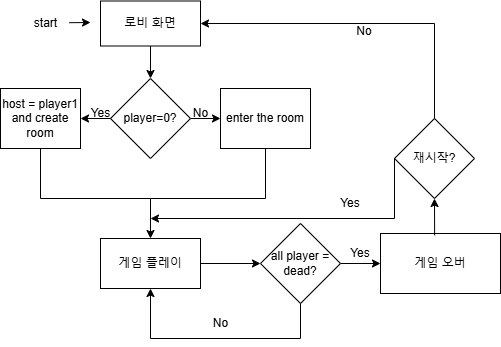
기존의 좀비 서바이벌 슈팅 멀티 게임에서는 자원의 관리, 반복적인 전투, 고립된 환경에서의 게임 진행이 특징입니다. 하지만 대부분의 좀비 AI가 예측 가능하며, 단순한 패턴이 고립된 환경과 맞물려 게임의 흥미를 떨어뜨릴 수 있습니다. 이런 문제점을 해결하고자 AI의 동적인 행동을 중심으로 게임을 설계하여, 플레이어가 게임의 흥미를 잃지 않으면서도 재미를 얻고자 합니다.

1. 프로젝트 목표

좀비 AI에 ‘퍼지 논리’를 적용해 플레이어가 플레이하면서 항상 도전적으로 진행하여 게임의 흥미를 유지하게 시키고 고양감을 얻게 합니다.

1. 프로젝트 설계

3.1 프로젝트 플로차트



<그림 1> 프로젝트 전체 플로차트

본 플로차트는 프로젝트의 전체적인 흐름을 나타냅니다.

(1) 프로젝트 실행 시 로비 화면이 등장합니다.

(2) 게임 시작 버튼 클릭 시 플레이어 인원이 없다면 플레이어1을 방장으로 하는 방을 생성하고 그 외의 플레이어들은 게임 시작 시 방에 참가시키게 합니다.

(3) 게임 플레이 도중 모든 플레이어가 사망하면 Game Over 씬을 로드하고 만약 그렇지 않았다면 계속 플레이를 유지합니다.

(4) Game Over 씬에서 방장이 Retry 버튼 클릭 시 다시 초기 게임 메인 화면으로 넘어가고 그렇지 않았다면 로비화면으로 넘어갑니다.

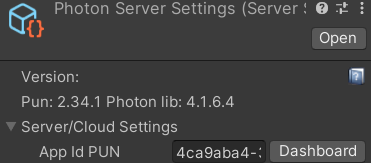
3.2 프로젝트 '퍼지 논리' 알고리즘 설계

|  |  |
| --- | --- |
|  |  |

<표 1> 퍼지 논리

<표 1> 기존의 좀비 서바이벌 슈팅 멀티 게임에서 플레이의 단조로움을 없애고자 작성한 코드로 퍼지화 모델은 맘다니형으로 제작하였습니다. <표 1>로 인해 Zombie.cs에서 플레이어와 거리가 멀어질수록 속도가 빨리지고, 가까워질수록 속도가 느려지는 조건을 제시하고 거리와 속도의 범위를 조정해 좀비 AI의 행동이 보다 긴장감을 높여주고 게임의 몰입감을 높여줍니다.

3.3 프로젝트 멀티플레이 설계



< 사진 1> Photon 서버

Unity 엔진과 Photon 서버를 이용해서 멀티플레이를 구현했습니다. Photon 네트워크를 사용해 클라이언트(Player Side)의 입력 및 체력, 아이템의 획득 여부, 좀비 AI상태를 서버로 동기화시키고 서버는 받은 데이터를 모든 클라이언트가 동일하게 확인할 수 있도록 전송하고 게임 오버의 조건을 달성하면 게임 오버 씬을 각 클라이언트에 전달해 로드합니다.

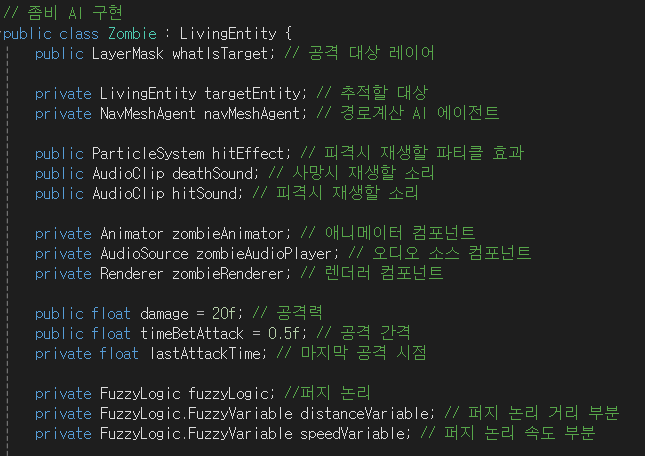
4. 프로젝트 구현

4.1 프로젝트 사용 기술

본 프로젝트에는 Unity 2021.2.15f1, Photon Unity Networking(PUN), 퍼지 논리 알고리즘, Visual Studio룰 사용했습니다.

4.2 프로젝트 상세 기술

4.2.1 좀비 AI 시스템



< 사진 2> 좀비 AI 코드

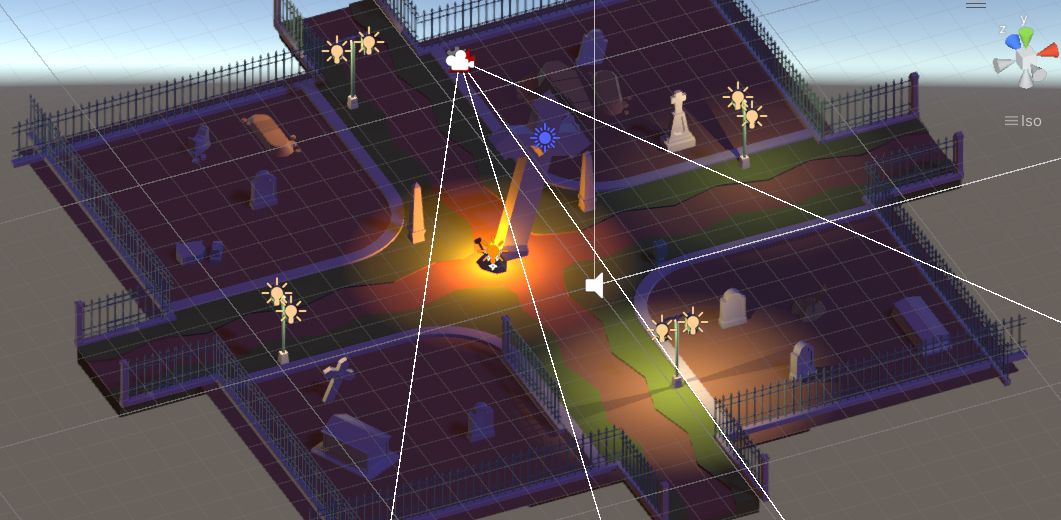
|  |
| --- |
|  |
|  |

<표 2> 퍼지 논리 코드

<사진 2>는 Unity의 NavMesh를 활용하여 플레이어를 추적하도록 경로를 계산하고,

<표 2>는 경로를 계산하고 이동할 때 플레이어와의 거리에 따라 자연스럽게 속도를 조절할 수 있도록 퍼지 논리를 작성했습니다. 본 코드에서는 플레이어와의 거리가 멀어질수록 플레이어에게 다가오는 속도가 빨라지고, 반대로 가까워지면 속도가 느려집니다.

4.2.2 프로젝트 맵 디자인



<사진 3> 프로젝트 맵 디자인

게임의 콘셉트와 어울리게 아포칼립스 느낌이 나도록 맵을 디자인하였습니다. 십자가 모양의 오브젝트 양 끄트머리 쪽에 좀비 AI 생성 구간으로 설정했습니다. 게임의 자원의 경우, 맵 전체에 생성 구간으로 설정했지만 생성되는 아이템의 반경은 플레이어 주변 반경으로 지정해 아이템이 플레이어의 현재 위치에서 생성되도록 하였습니다.

4.2.3 프로젝트 멀티플레이 기능 구현

|  |
| --- |
|  |
|  |

<표 3> 포톤으로 구현된 방 생성 코드

<표 3>에서 생성된 마스터 서버에서 게임 시작시 실시간으로 참가하는 플레이어들의 상태를 동기화하여 각 클라이언트에게 전달해줍니다.

4.2.4 프로젝트 아이템 기능 구현

|  |  |
| --- | --- |
|  |  |

<표 4> 프로젝트 아이템 관련 사진 및 코드

<표 4>와 같이 본 프로젝트에는 아이템 총 3개 있습니다.

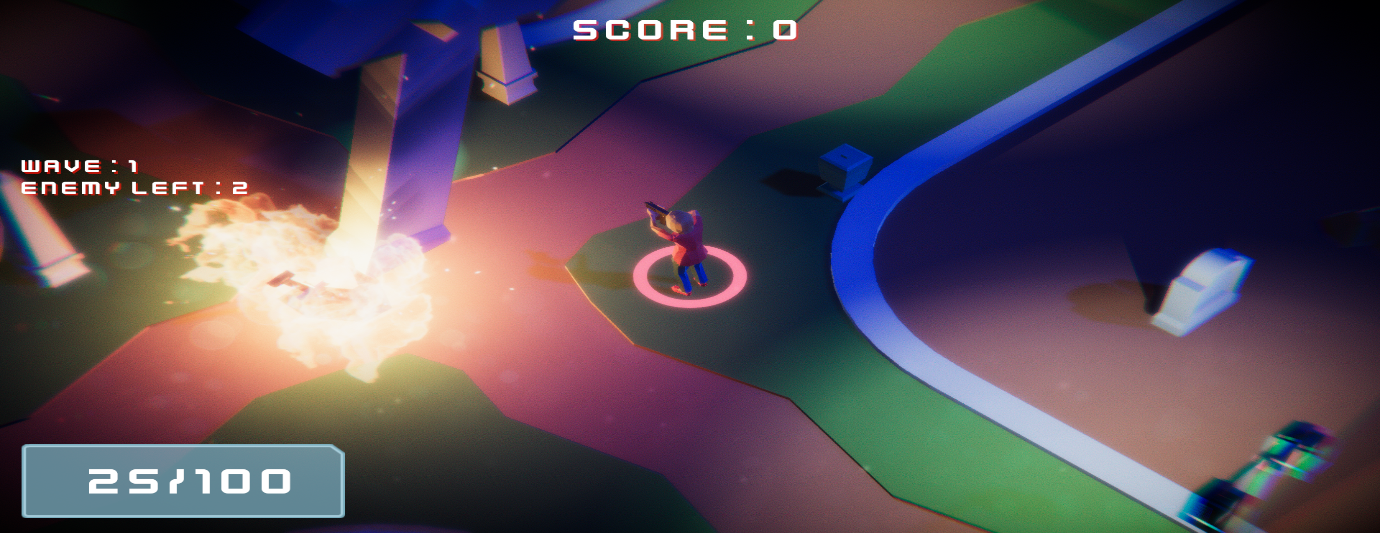
(1). 플레이어의 체력을 회복시켜주는 아이템(하트 모양의 오브젝트)

(2). 플레이어의 탄환을 보급시켜주는 아이템(탄환 모양의 오브젝트)

(3). 플레이어의 점수를 증가시켜주는 아이템(코인 모양의 오브젝트)

관련 아이템들은 Phothon 서버에서 관리해 생성과 소멸을 각 클라이언트에 전달하고 표시합니다. 또한, 플레이어들 중 한 명이 아이템을 습득하면 습득한 아이템의 효과에 따라 증가하고 동기화 후 다른 클라이언트들에게도 증가한 수치를 보여주도록 실행합니다. 그리고 습득한 아이템은 습득 즉시 소멸됩니다. 아이템의 충돌 처리는 sphere collider를 사용했습니다.

4.2.5 프로젝트 UI 구현



<사진 4> 프로젝트 게임 화면 UI

전체적인 UI는 게임 플레이에 방해되지 않는 선에서 최소한으로 구현했습니다.

좌측의 상단과 하단에 현재 웨이브, 남아 있는 좀비 수, 현재 탄약 / 전체 탄약을 숫자와 최소한의 단어로 구현하고, 중간 상단에 현재 점수를 표현하는 UI를 넣어 게임 플레이와 가까운 장소에 배치해 실시간으로 확인해도 게임 플레이에 영향이 가지 않게 끔 구현했습니다.

4.2.6 프로젝트 플레이어 캐릭터 구현

|  |  |
| --- | --- |
|  |  |

<표 5> 플레이어 캐릭터 입력과 움직임

Unity의 Animator를 사용하여 이동 및 공격 애니메이션을 설정했고, Rigidbody 컴포넌트와 Capsule Collider 컴포넌트를 적용시켰습니다. 캐릭터는 키보드 입력을 통해 'W'와 'S' 키는 전진과 후진으로 이동, 'A'와 'D' 키는 왼쪽 회전과 오른쪽 회전을 하게 만들었습니다. 또한, 마우스 왼쪽 버튼 클릭 시 공격을 실행하도록 했습니다.

4.2.7 프로젝트 플레이어 무기 구현

|  |  |
| --- | --- |
|  |  |

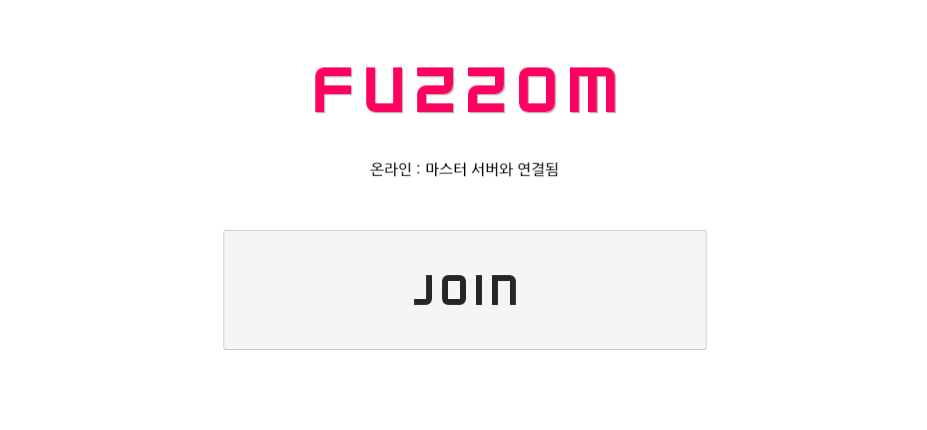
<표 6> 플레이어 무기 상태와 퍼지 논리

플레이어의 무기는 SMG로, 마우스 공격 버튼을 한 번씩 누를 때마다 한 발씩, 입력을 유지하면 연사가 되도록 구현했습니다. 탄환은 Raycast를 사용해 타격 판정을 계산하며, 적과 충돌 시 피격 이펙트가 발생하도록 하였습니다. 플레이어는 한 번에 25발의 탄환이 주어지며, 탄환이 적어도 하나는 비어있어야 재장전이 되도록 구현했습니다.

재장전 버튼은 'R'키로, 수동으로 눌러야 합니다.

5. 최종 결과물

5.1 메인 화면



<사진 5> 프로젝트 메인 화면

게임 구동시 서버와 연결하고 연결이 되었으면 '서버와 연결됨' 메시지를 출력합니다. 'Join' 버튼 클릭 시 게임에 참가하는데 만약 플레이어가 없으면 방을 생성 후에 참가하고 플레이어가 한 명이라도 있으면 그 방에 참가하게 됩니다.

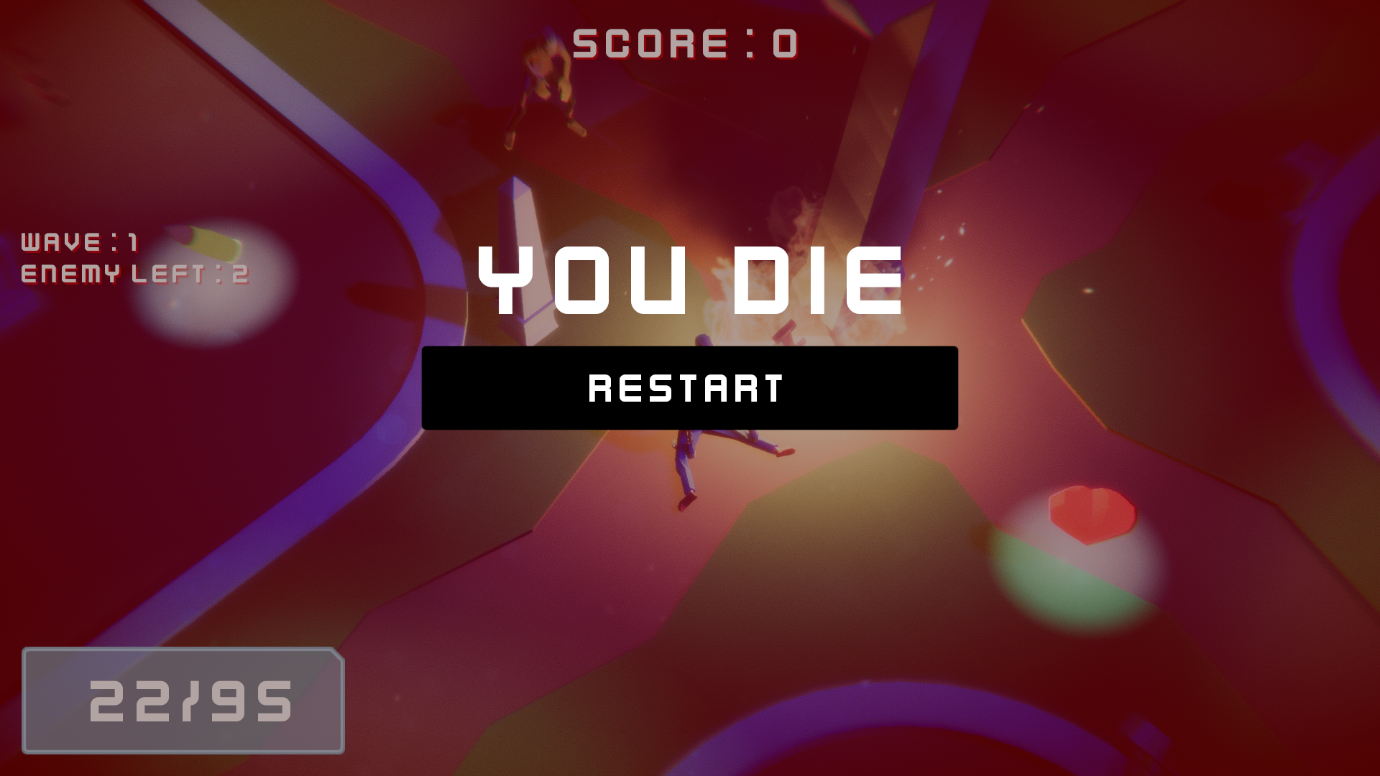
5.2 게임 플레이 화면



<사진 6> 게임 멀티플레이 화면

게임에 참가한 플레이어들은 실시간 클라이언트-서버 통신으로 관리되는 좀비AI들로부터 아이템을 습득하고 최후까지 생존하여 점수를 얻으면 됩니다.

5.3 게임 오버 화면



<사진 7> 모든 플레이어 사망시 출력되는 게임 오버 화면

솔로플레이 또는 멀티플레이 여부에 상관없이 게임 룸에 참가한 모든 플레이어가 사망하게 되면 GameManager를 통해 GameOver Scene을 로드하도록 합니다. 'Restart' 버튼은 최후까지 살아남은 플레이어가 사망할 시에만 등장합니다. 'Restart'버튼을 누르면 모든 플레이어가 웨이브 1부터 다시 시작합니다.

6. 문제점 및 개선방안

6.1 문제점

1. 플레이어가 아이템을 계속 습득하여 시간을 끌 때의 페널티가 없어 게임의 흥미를 떨어뜨릴 수 있습니다.

6.2 개선 방안

1. 시간을 끄는 행위에 제지를 가하는 방법이 여러 개 있습니다.

(1) 웨이브 시작하고 특정 시간이 지나면 아이템 효과의 효율을 점점 줄이는 방식

(2) 일정 시간이 지나면 좀비AI의 능력치(체력, 대미지, 속도)를 증가시키는 방식

(3) 좀비AI가 2마리 이상이면 일자로 추격하는 것이 아닌 포위하는 그물 형태로 추격하는 방식

좀비AI 또는 아이템, 시간을 추가하는 방식으로 위의 문제점을 해소할 수 있을 것 같습니다.

6.3 기타 피드백

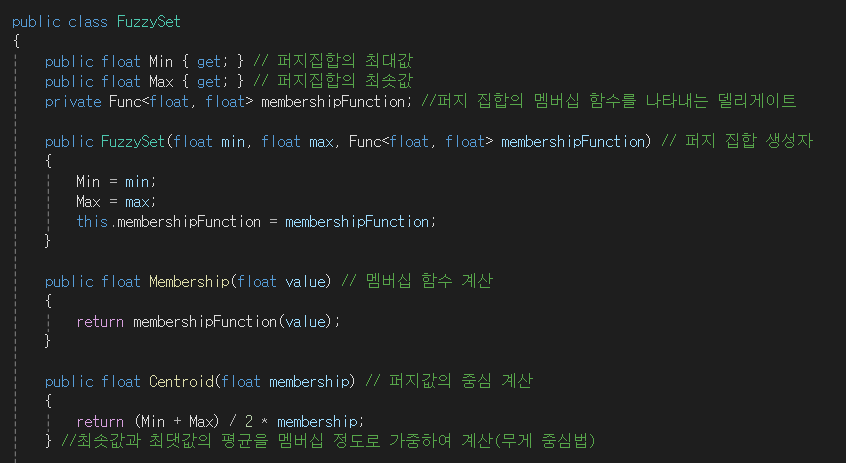
1. 각자 고유 총기를 사용하는 캐릭터 4명을 선택하는 시스템을 도입

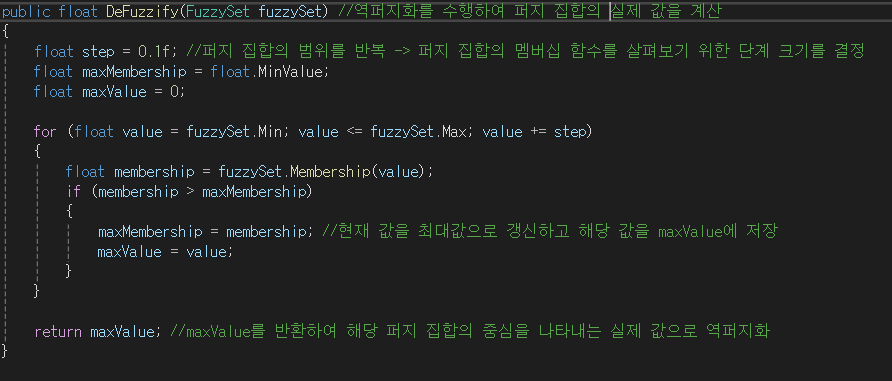
2. 총기를 3가지 더 추가하여 스왑 시스템을 도입

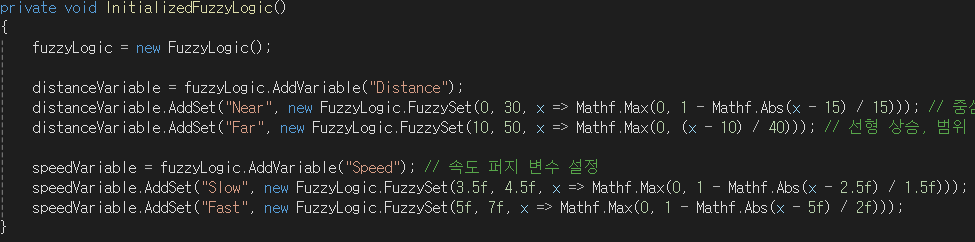
위의 피드백들이 졸업작품 전시회를 하면서 사용자들에게 받은 것입니다.

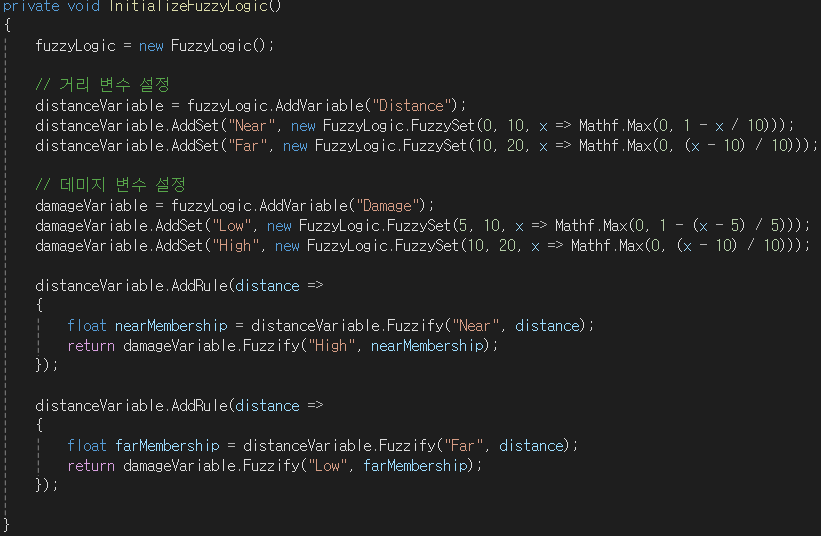
7. 코드

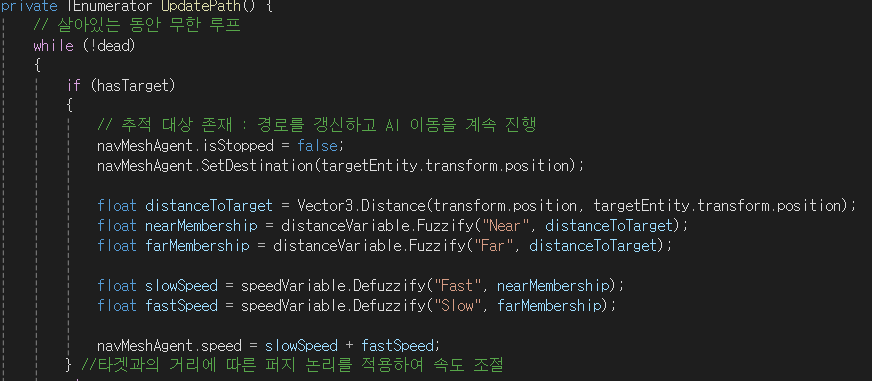
7.1 퍼지 논리



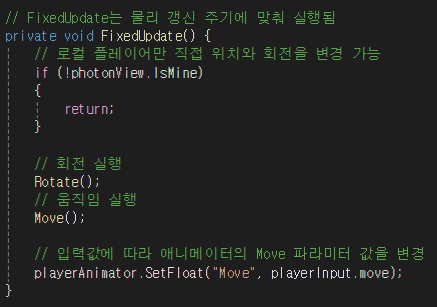


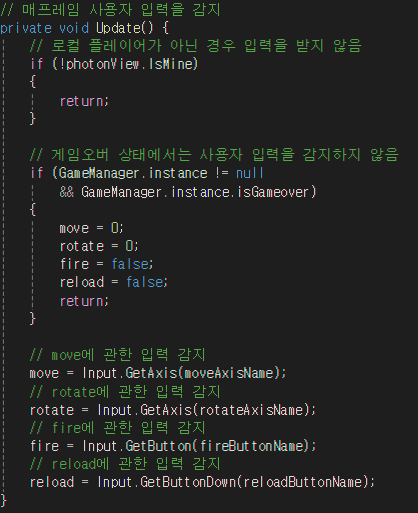


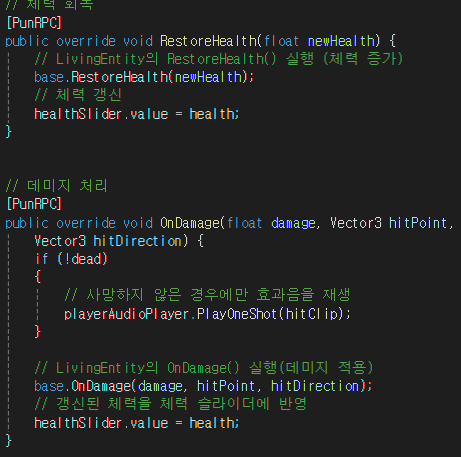


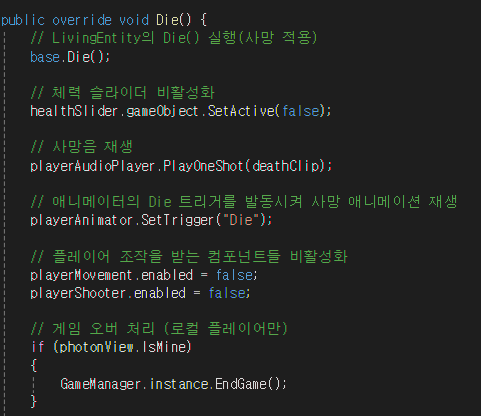


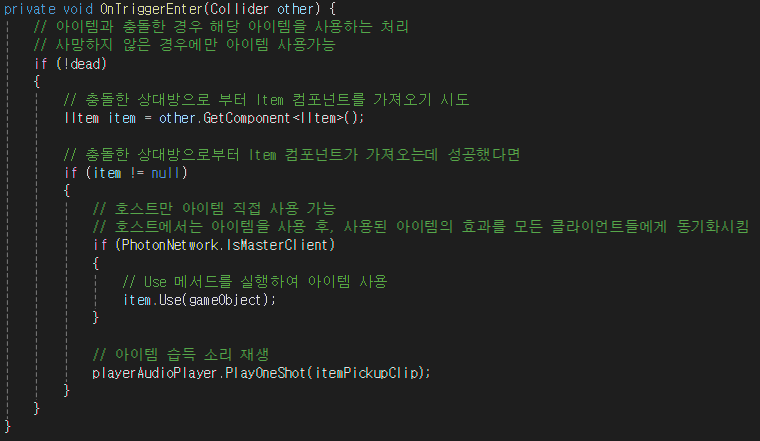
7.2 Photon 관련 코드

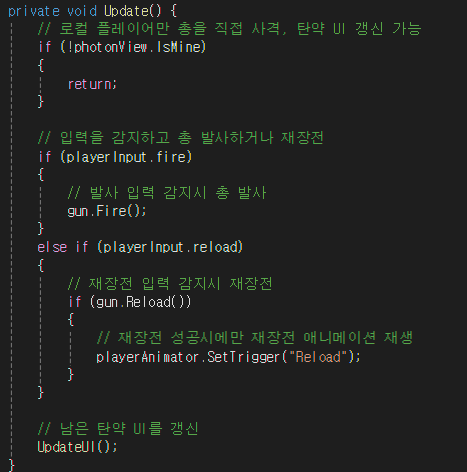


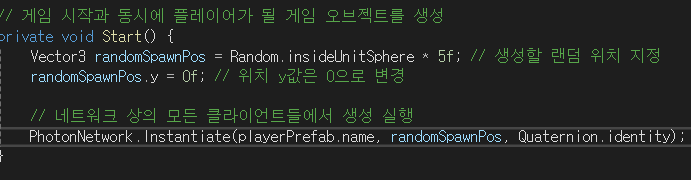


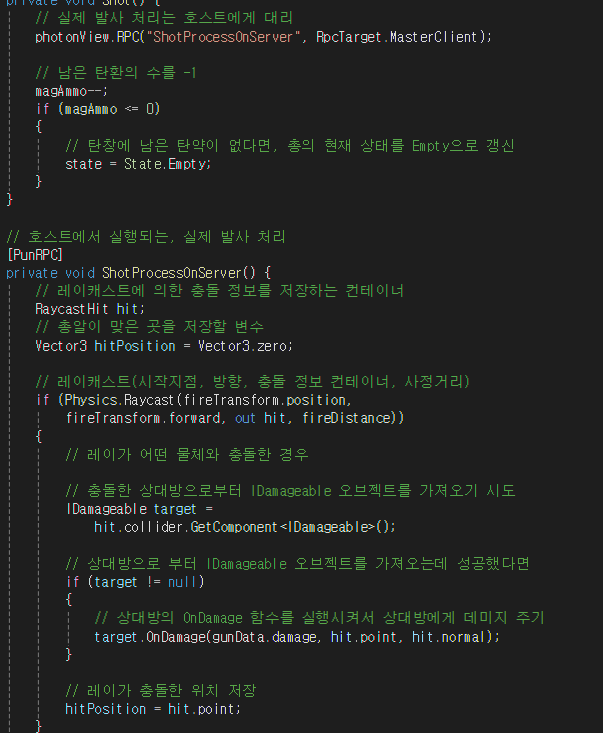


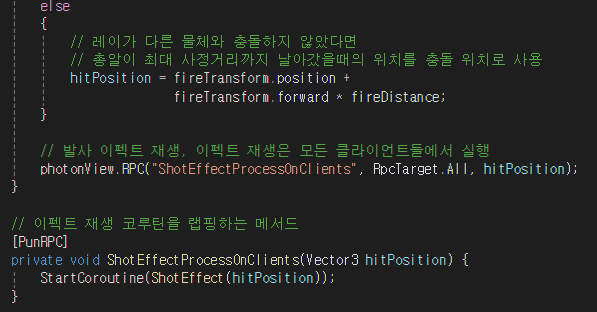


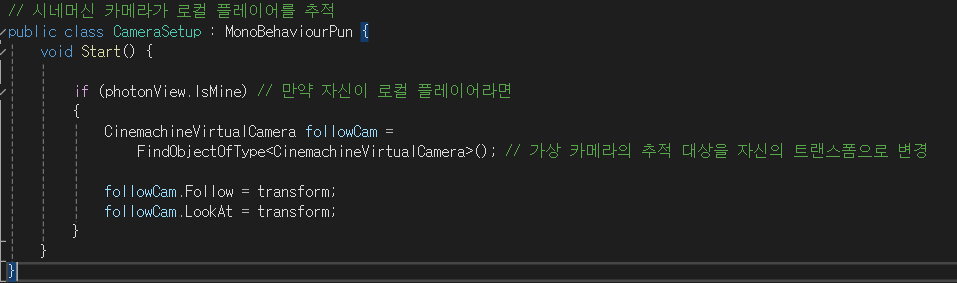


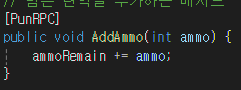


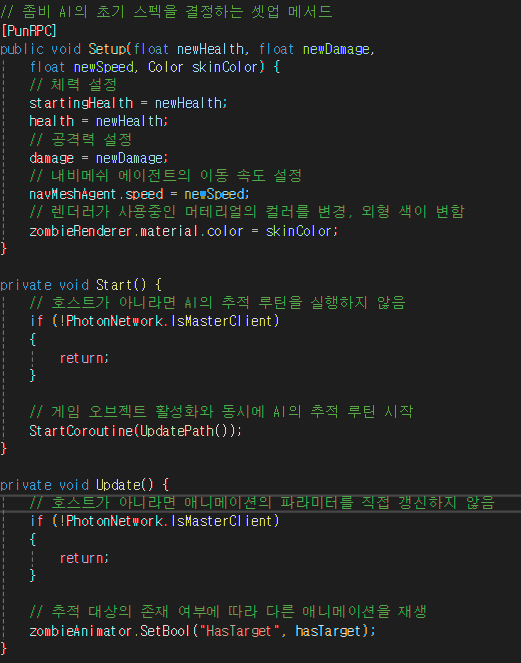


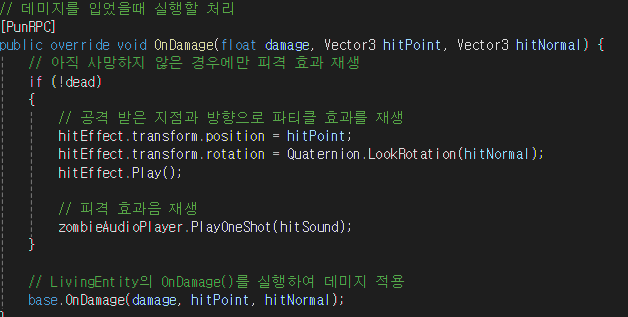


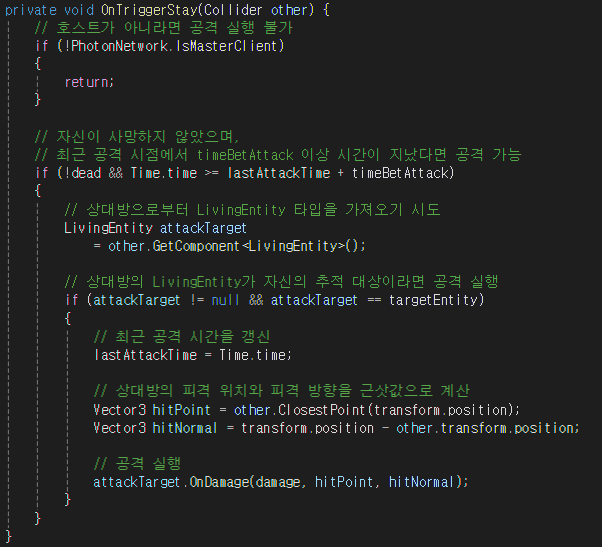


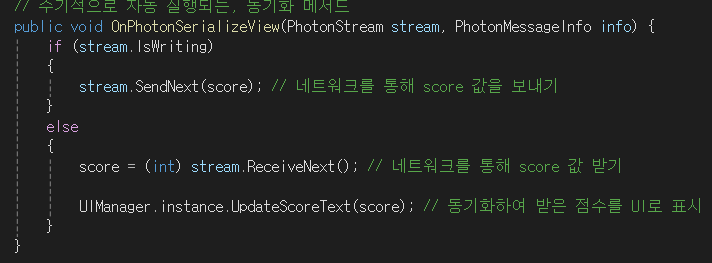


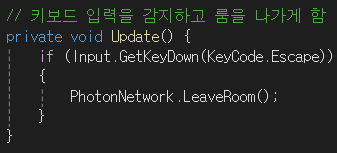


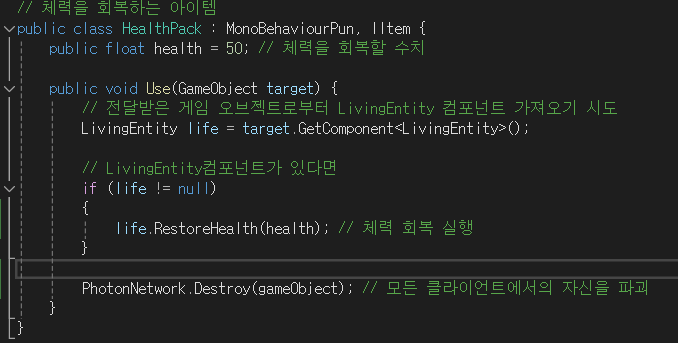


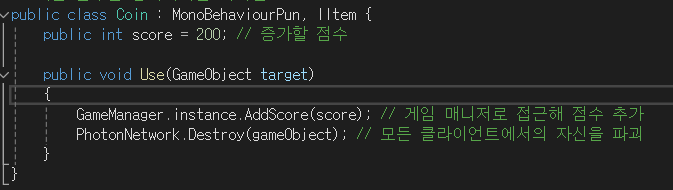


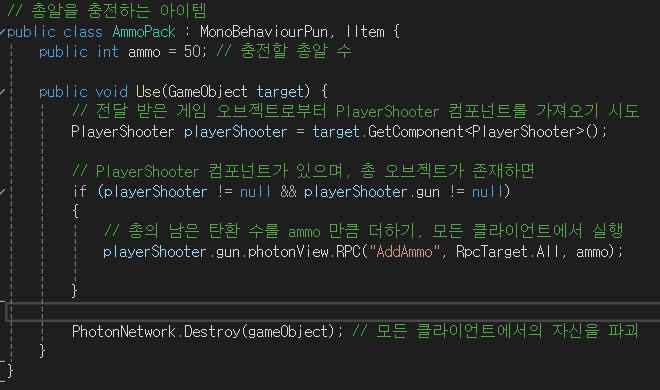


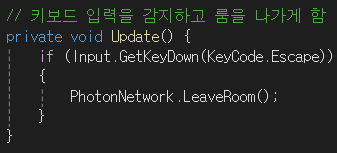


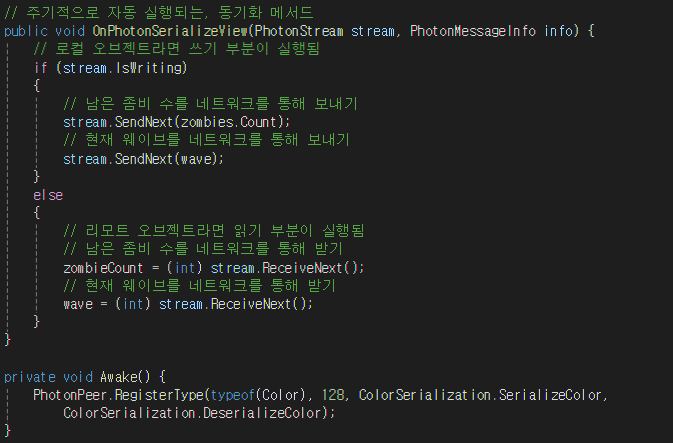


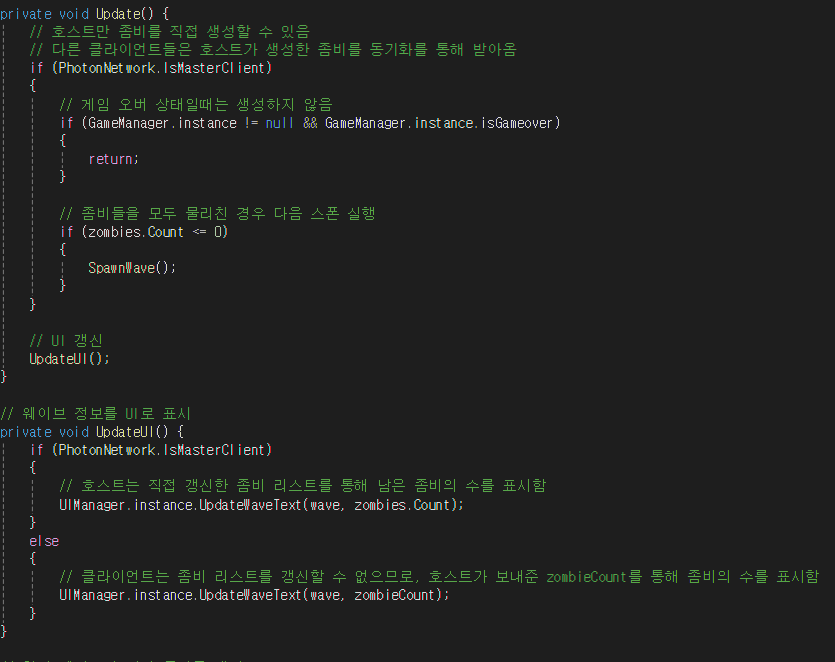


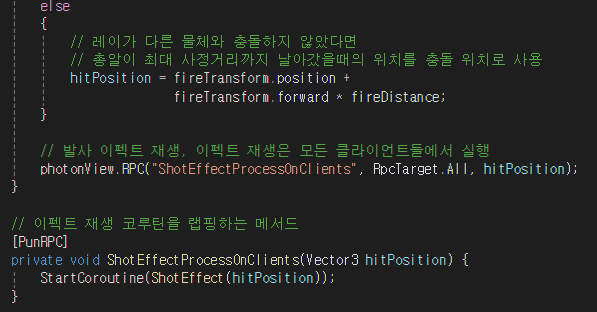




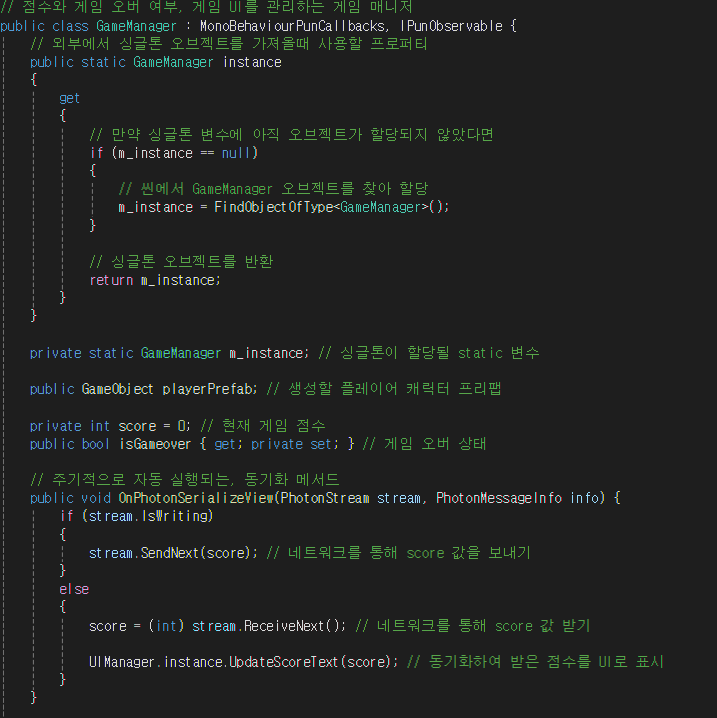


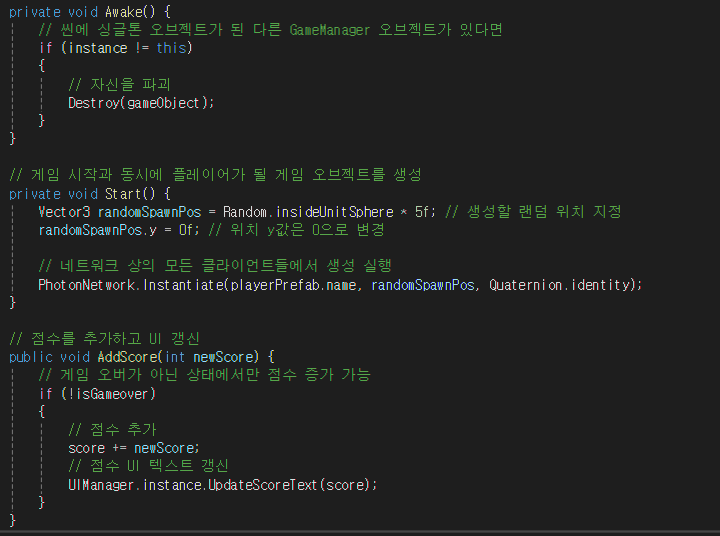


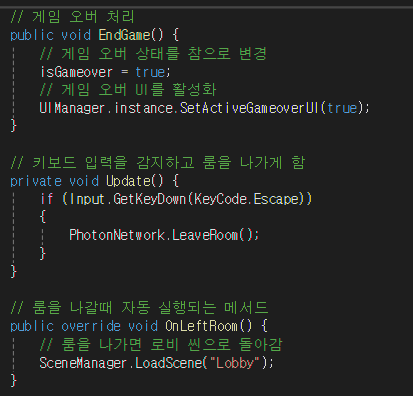




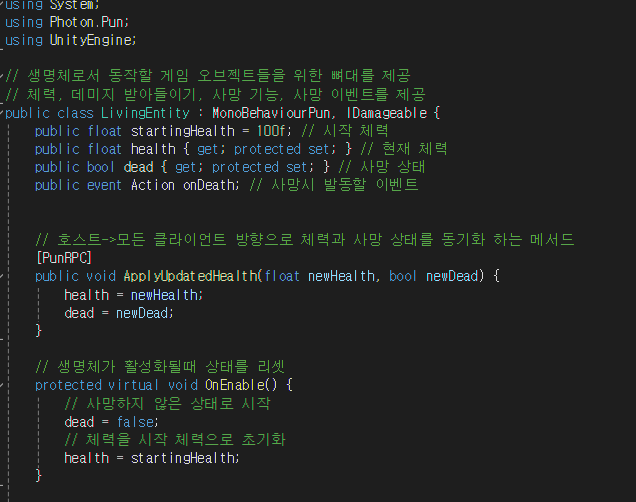
7.3 GameManager

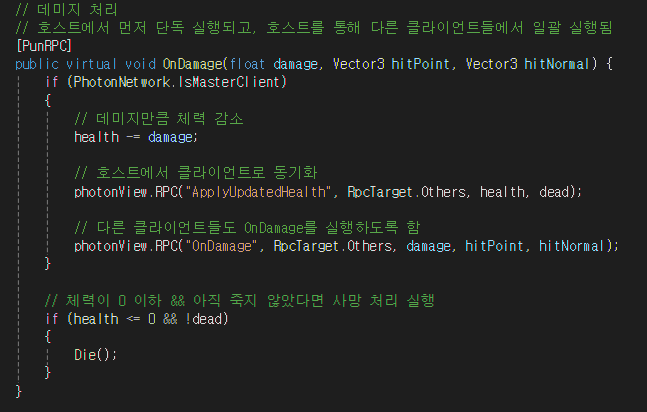


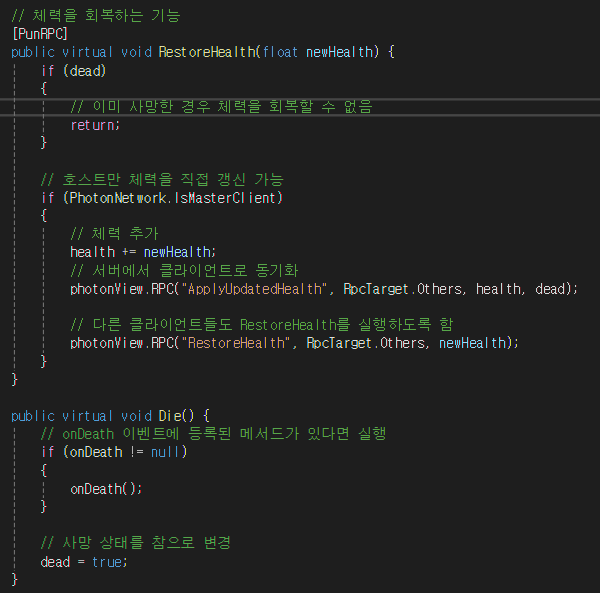




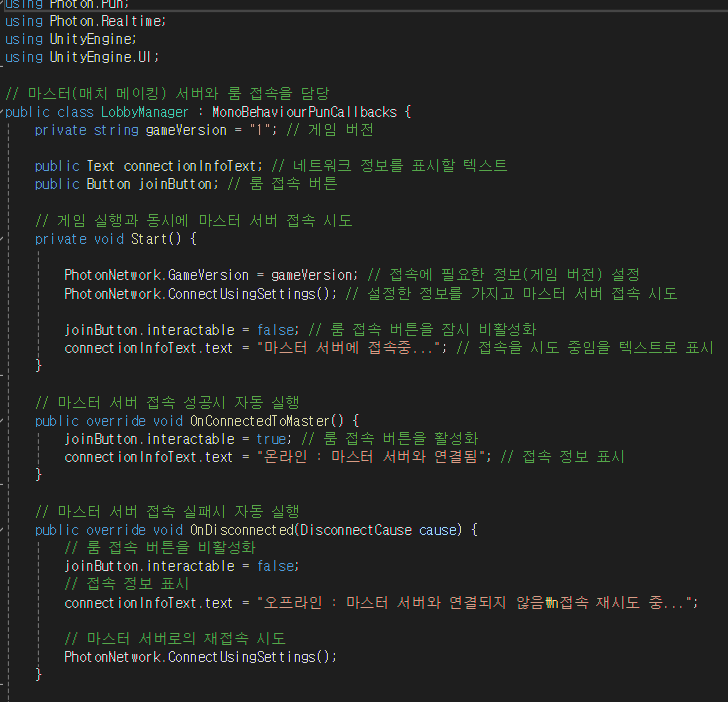
7.4 LivingEntity

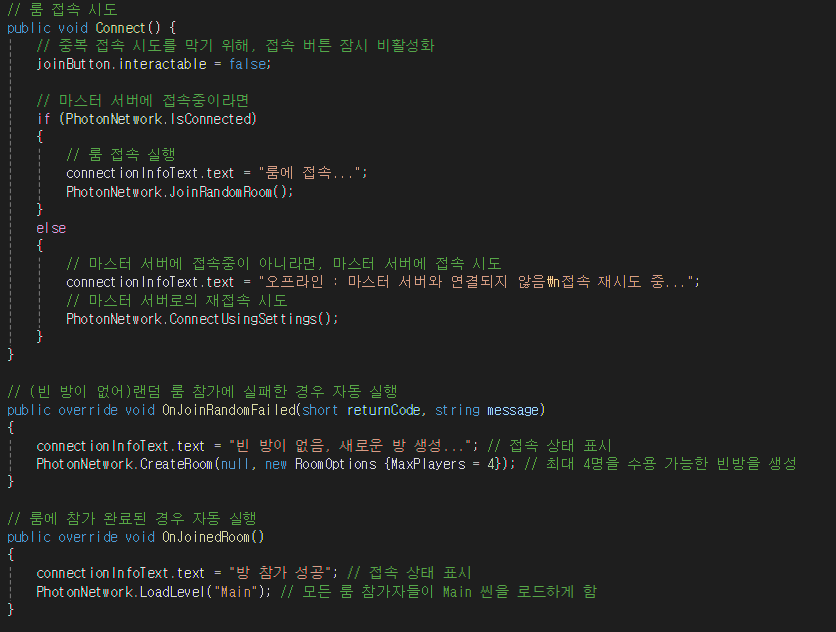




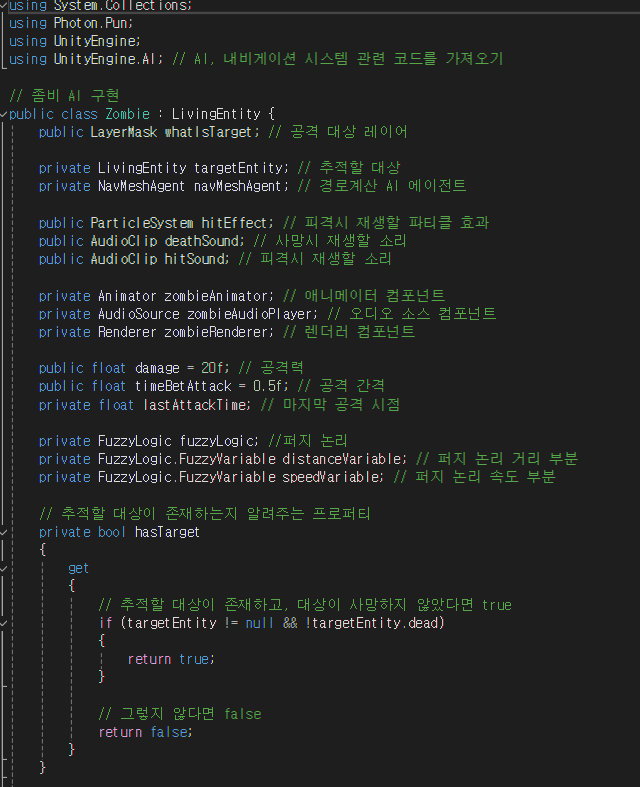


7.5 LobbyManager

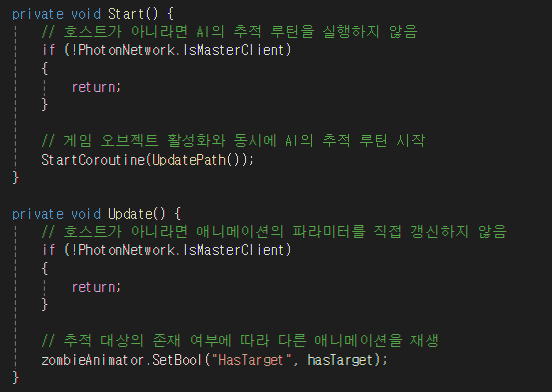


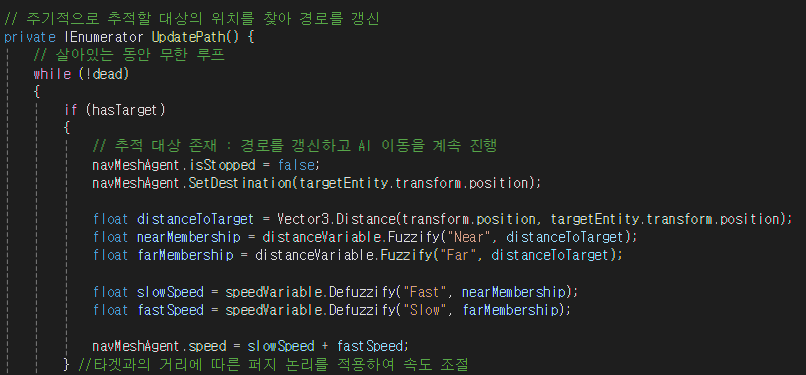


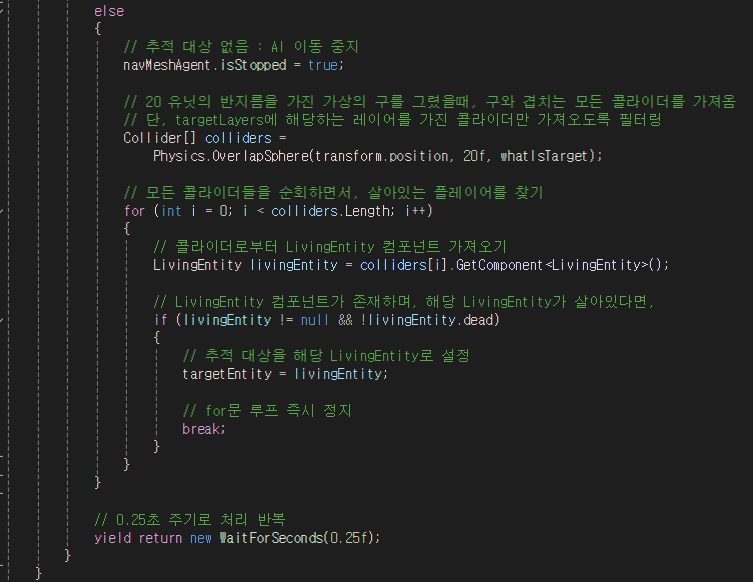
7.6 좀비AI

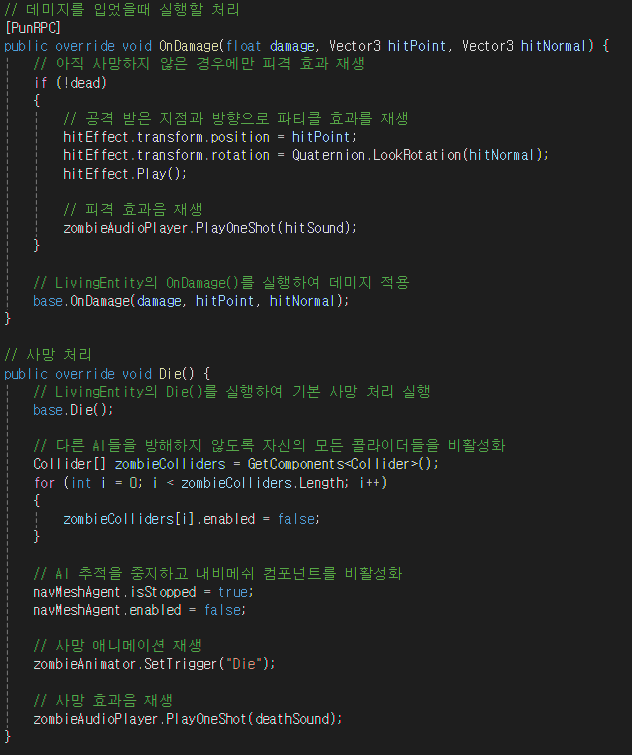


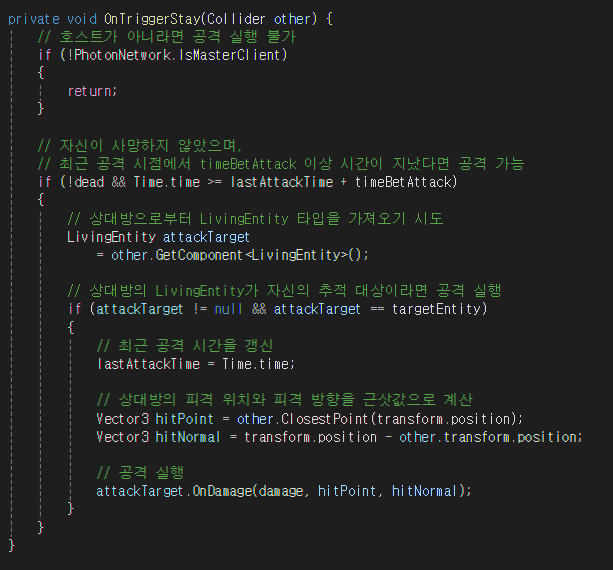




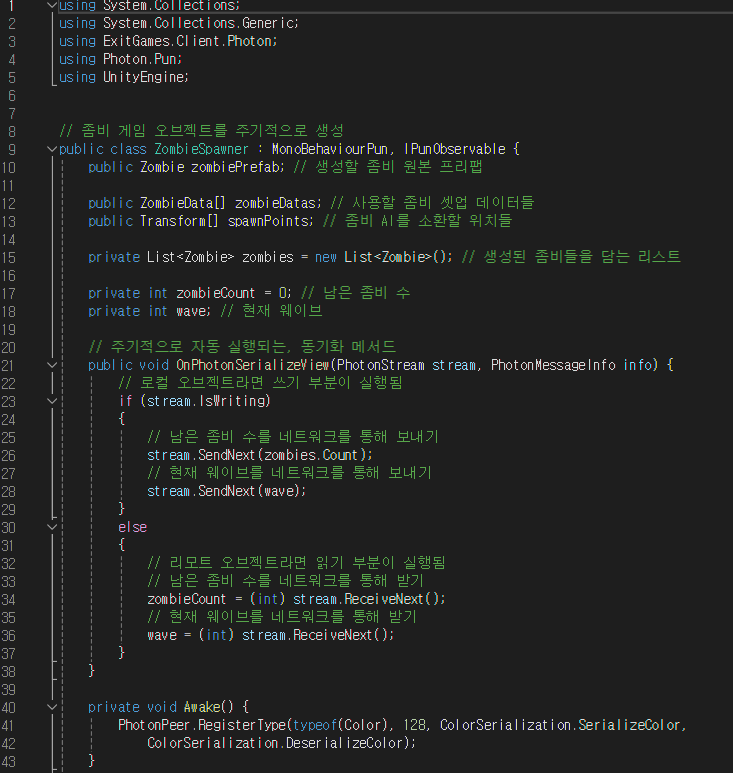


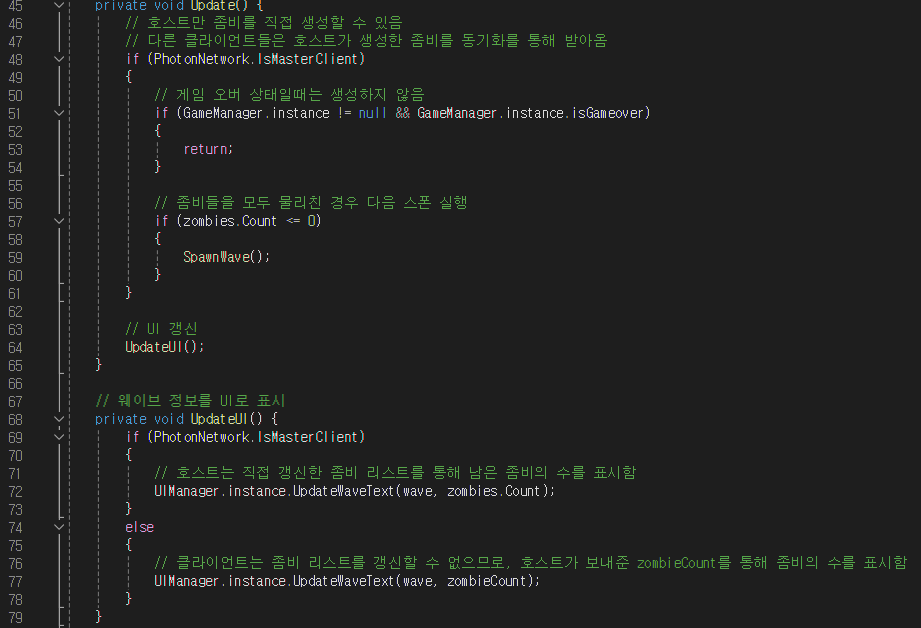


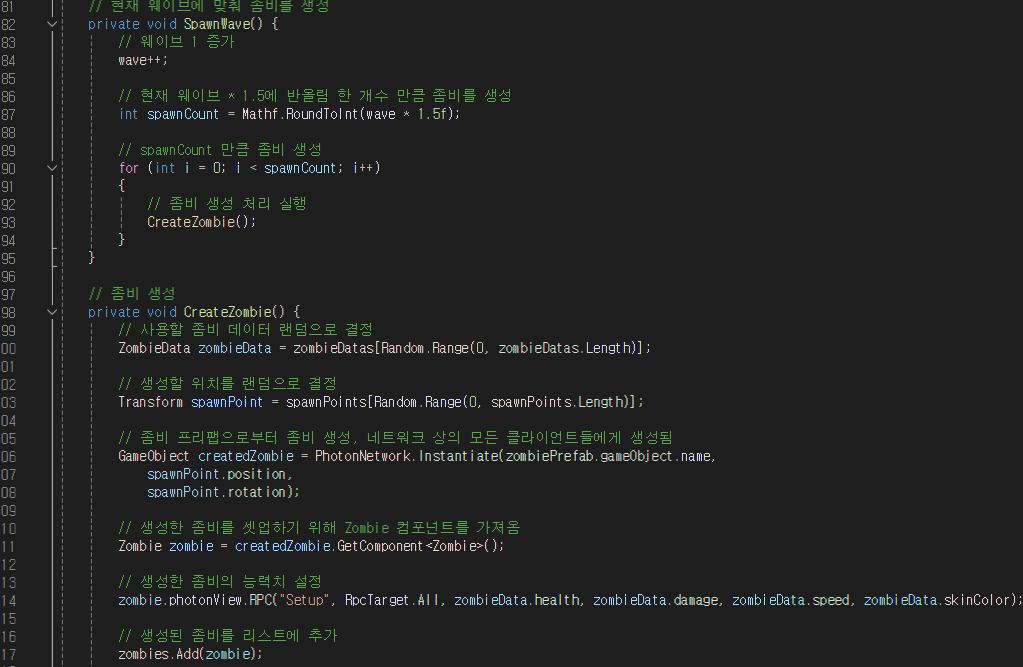


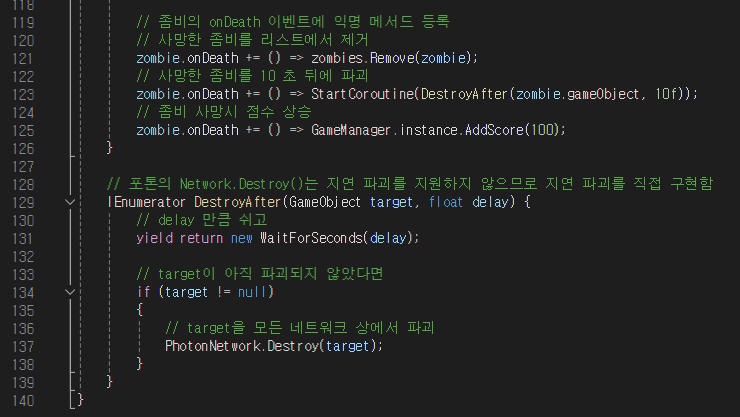


7.7 좀비AI 스폰









8. 참고문헌

[1] 유니티 공식 사이트 [Unity 사용자 매뉴얼 2021.3(LTS) - Unity 매뉴얼](https://docs.unity3d.com/kr/2021.3/Manual/UnityManual.html)

[2] 포톤 공식 사이트 [Pun 2 소개 | Photon Engine](https://doc.photonengine.com/ko-kr/pun/current/getting-started/pun-intro)

[3] c# Fuzzy Logic [GitHub - chengkehan/FuzzyLogic: A fuzzy logic implementation with c# and friendly ui in Unity.](https://github.com/chengkehan/FuzzyLogic)