**포트폴리오**

클라이언트 부문

한국산업기술대학교

게임공학과

장동필

2018.09.18

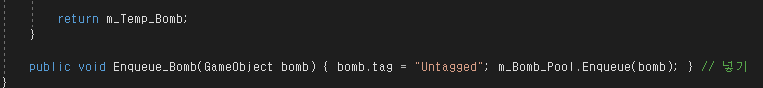
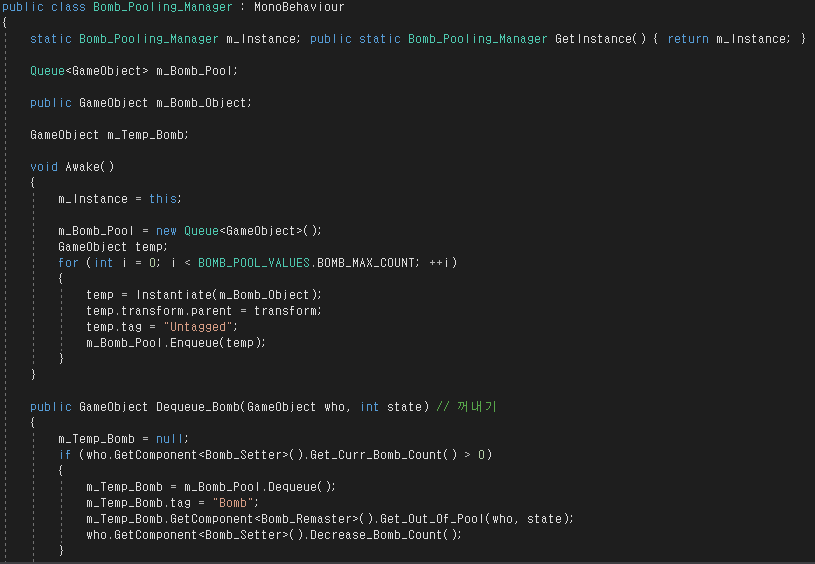
* 목차

1. Turtle Bomb
2. Penguin Duo
3. **Turtle Bomb**



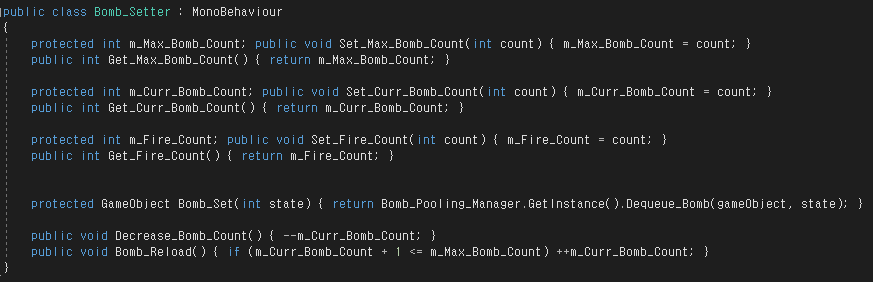
* **장르 :** 캐주얼 아케이드 (3인칭)
* **게임 소개 :** 거북이 (플레이어)가 폭탄으로 적 (침략자 고블린 무리들)을 물리치는 Bomber Man 류의 게임.
* **제작 환경 :** Unity 2017 engine (C#)
* **제작 기간 :** 약 1년 (2017.09 ~ 2018.09)
* **제작 인원 :** 3인
* **맡은 역할 :** 클라이언트 제작
* **구현 내용 :** ‘앱 시작’ ~ ‘모험 모드’ 까지의 기능 제작. (Player 조작, Monster AI, UI 기능, Effect 편집 및 적용, 아이템, 미션, … 구현)
* **핵심 소스 코드 :**

1. **폭탄 오브젝트 풀링**



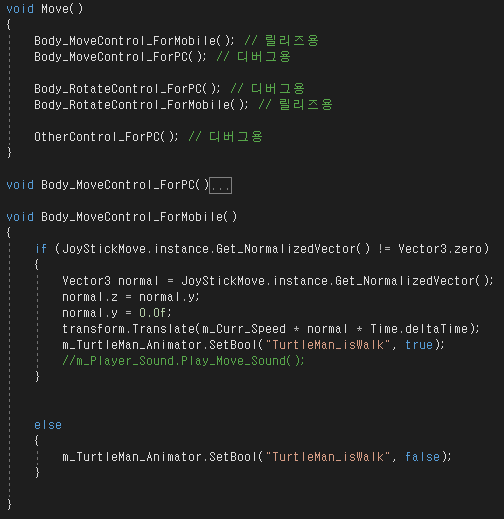
-> 싱글톤으로 구성하여 최초 씬 시작 시 정의한 폭탄 개수만큼 폭탄을 생성하여 큐에 저장. 이후 폭탄을 사용할 때 인스턴스를 통해 인큐/디큐 할 수 있도록 구현.

1. **Bomb\_Setter 상속**

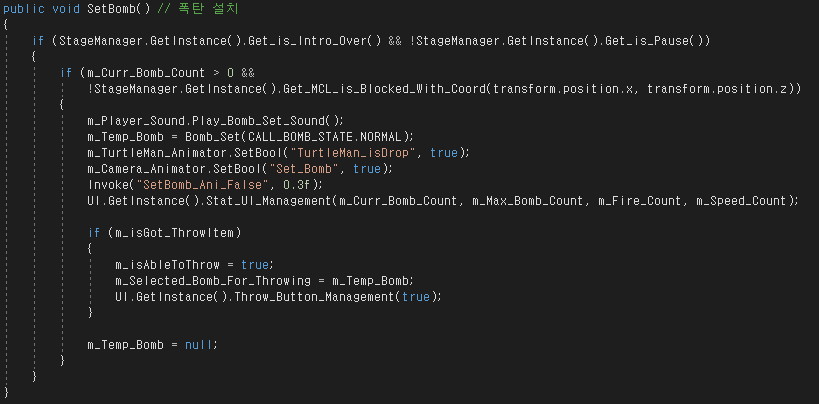


-> 폭탄을 설치할 수 있는 객체 (캐릭터)를 사용함에 있어서 그 기반이 되도록 하는 클래스인 Bomb\_Setter. 폭탄을 사용할 때 반드시 가져야할 데이터를 가지게 하며 폭탄 설치 시 ‘폭탄 풀링 매니저’를 부르도록 한다. 이후 이 클래스를 상속하여 새로운 캐릭터들을 제작하도록 한다.

-> Bomb\_Setter를 상속한 Player.

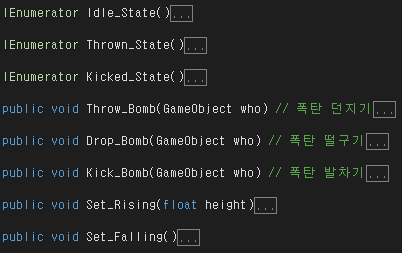


-> 조이스틱을 이용한 이동이라던지,

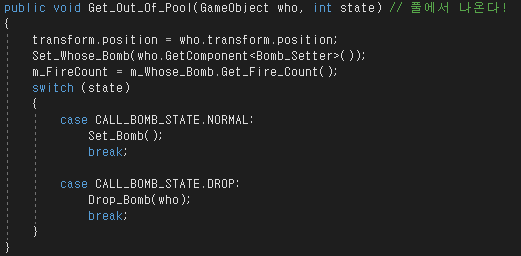


-> 폭탄을 설치할 때 수행할 것들을 정의하는 등 플레이어에게 필요한 기능을 추가적으로 다룬다.

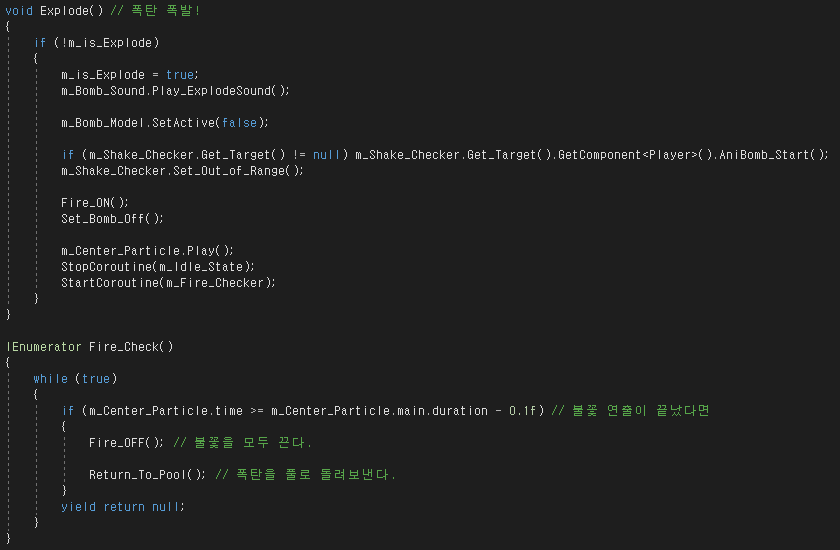
1. **폭탄**



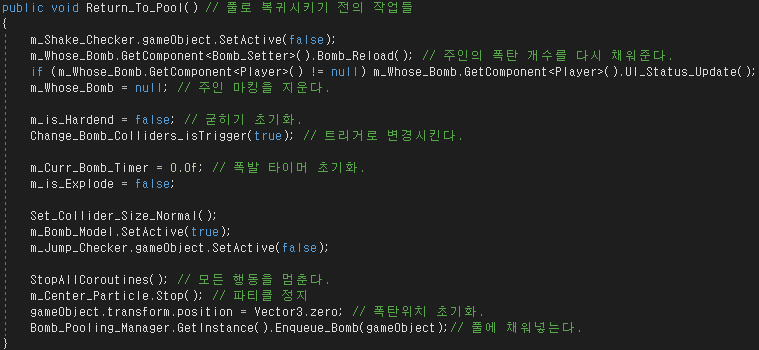
-> 폭탄은 위와 같은 상태들로 움직임을 표현한다. 상태는 항상 유일하게 작동한다. (FSM)



-> 풀링된 상태에서 맵 상에서 작동하는 상태로 전환하도록 하는 메소드로, 누가 설치했는지와 그 객체가 가진 정보 (화염 최대 길이, 위치 좌표, 시작 시 폭탄의 상태)들을 가져와서 설정해준다.

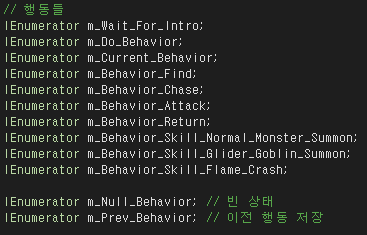


-> 폭탄이 폭발 후 풀로 돌려 보내기를 시도하기 까지 수행되는 메소드. Explode()는 idle 상태에서 일정 시간 초과 시 또는 다른 화염과의 충돌 시 호출된다. 이후 화염 이펙트를 실행시킨 후 끝나기 까지를 체크하여 풀로 돌려보낸다.

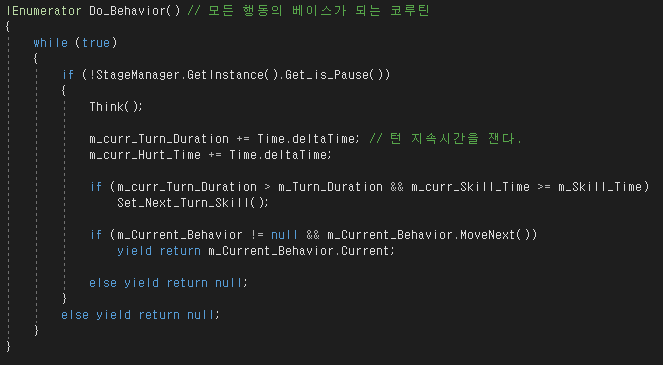


-> 풀로 복귀 하기 전 수행 할 작업들. (초기화 작업)

1. **보스 AI**



-> 보스의 행동들을 정의



-> Do\_Behavior()에서 현재 상태에 대한 유일한 행동을 수행한다.

Think() 메소드는 상태 전환에 도움을 주는 메소드 (상황에 맞는 상태변화) 이다.



-> 이 코드는 Attack() 메소드의 일부이다. 공격 시 애니메이션의 정규화 시간에 따라 추가적인 상태변화를 준다. 여기서는 공격 초반에 느린 공격 속도로 진행 및 공격 범위 표시기 활성화를 수행, 이후 중반에는 충돌체 까지 활성화하면서 공격 속도를 빠르게 바꿔주고 공격 모션이 완료된 시점에 위의 사항들을 비활성화 시킨다.

* **제작 난관 :**

1. 유니티를 사용함으로써 **디버깅**하는데 소비되는 시간이 상당히 길었고 난잡했다. 때문에 자연적으로 기능 구현에 드는 시간이 증가했다.
2. 제작 초기에 C#과 유니티의 다양한 기능들을 모른 채로 제작했던 것들을 리뉴얼하는 과정에서 난관을 겪었다.
3. 팀원들 간에 작업 시간 조율이 잘 이루어지지 않아 모두가 모여서 작업할 시간이 많지 않았다.
4. 모바일 환경에서의 최적화 문제. (+ 그래픽 퀄리티 vs FPS(성능) 딜레마)

* **난관 극복 :**

1. 상당히 애먹은 부분이다. 로그를 찍는 것 또한 가독성이 떨어져서 UI 텍스트 상에 표시하는 방법을 주로 사용했다.
2. 제작 초기의 기능들은 스파게티 식으로 얽혀 있어 일부 객체가 빠지면 에러를 뱉어냈다. 이들을 더 가볍고 객체지향적으로 구현하여 객체들 간에 큰 영향을 주지 않도록 하였다.
3. 기획자의 주도 하에 각자 개인적으로 할 일을 부여하였고, 일정에 맞게 해결 함으로써 극복하였다. 또한 특이사항은 카카오톡과 깃허브를 통해 전달하였다. 기존에 해결되지 않던 것들은 기록해 두었다가 한자리에 모여 개발을 진행할 때 빠르게 처리할 수 있도록 하였다.
4. 모바일 환경에서 최적화가 절실한 것이 사실이었다. 공모전 전시 당시 사용한 모바일 기기는 최신형이라 일체의 성능저하 없이 게임이 원활하게 진행되었으나, 사실 2~3년 이상 지난 기기에서는 성능저하가 눈에 띄게 일어났다. 초기 개발 당시의 그래픽 퀄리티에 비해 현재의 그래픽 퀄리티는 확연히 낮은 버전으로 빌드를 하고 있음에도 불구하고 성능저하가 일어났다. 더 이상 퀄리티를 낮추고 싶지 않아서 최대한 배치 수를 줄이고 인스턴스 생성/해제를 최대한 줄이는 방향으로 리뉴얼 했으나 성능에는 큰 차이가 없었다. 이유는 기본적으로 배치 수가 상당하기 때문이라고 추정했다. 때문에 개발지원 수상자에 대한 멘토링을 통해 배치 최적화 방법을 알아내려고 한다.

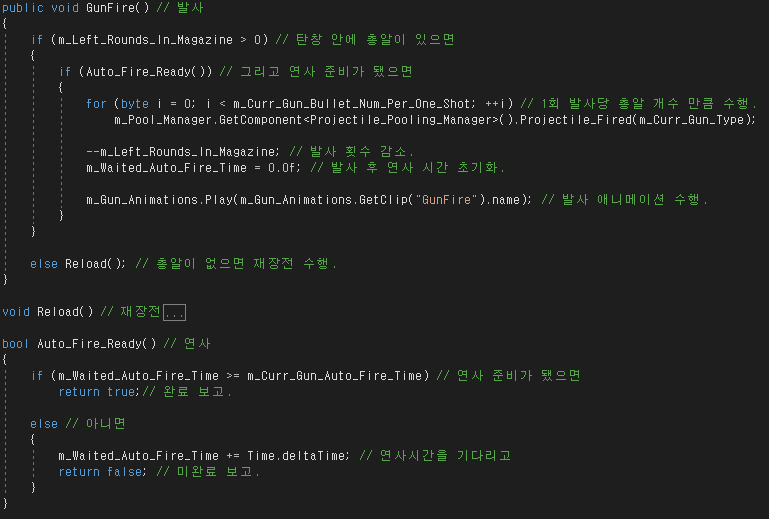
* **수상 및 전시 :** 2018성남 인디게임 공모전 개발지원 부문 우수상 수상 및 전시, 2018 산업기술대전 전시
* **소개 영상 :** <https://www.youtube.com/watch?v=lQ6Y9ieEmNY>

1. **Penguin Duo**



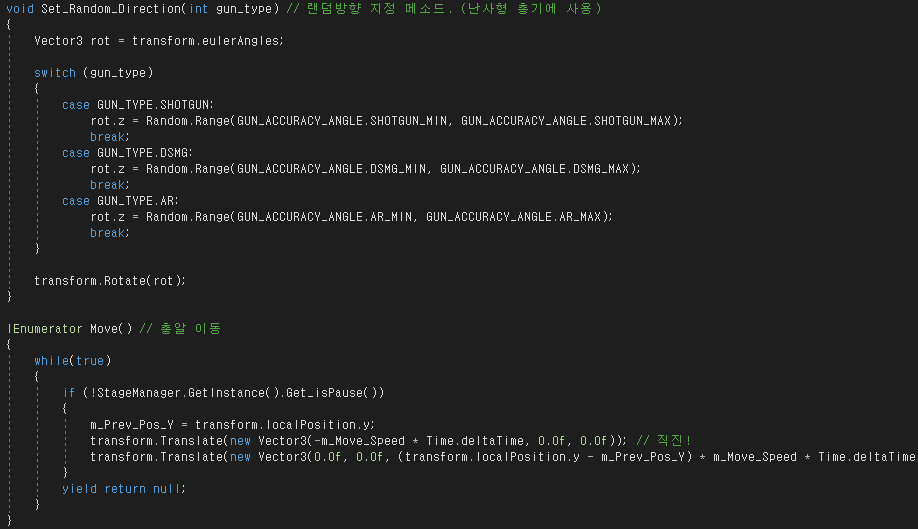
* **장르 :** 2.5D 횡스크롤 레이싱. (2.5D라 표현한 이유는 2D 그래픽 게임이지만 사실상 오브젝트들이 3차원 상에서 움직이고 있기 때문.)
* **게임 소개 :** 녹아버린 남극의 빙산 위에서 포식자들에게 쫓기는 두 마리의 펭귄 형제들이 적을 물리치며 한없이 도망치는 레이싱 게임.
* **제작 환경 :** Unity 2017 engine (C#)
* **제작 기간 :** 약 2주 (2018.7~2018.8)
* **제작 인원 :** 1인
* **맡은 역할 :** 클라이언트 제작 및 리소스 제작.
* **구현 내용 :** 게임에 사용된 모든 비주얼 리소스 그림 (제작), 플레이어 기능 제작 (총기류 발사, 이동, 사망, …), 몬스터 별 AI 제작, 장애물 제작, 아이템 제작, 데이터 테이블 (패턴 테이블) 제작, UI 제작, …
* **핵심 소스 코드 :**

1. **총기**



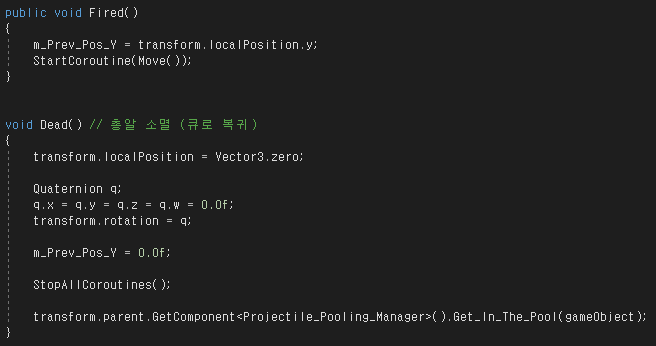
-> 총기의 발사 과정. GunFire() 메소드가 주축이 되며, 이는 Player의 Attack() 메소드에서 호출된다. 발사 조건 충족 시 풀링된 투사체를 발사할 개수 만큼 꺼내 투사체에게 발사 명령을 내린다.

1. **투사체**



-> 투사체가 꺼내졌을 때 랜덤 방향으로 이동하도록 하게 해주는 메소드. 각각의 총기의 특성에 따라 랜덤 범위가 달라진다.

지정된 방향에 따라 투사체는 직진을 하게 되며 또한 Y좌표에 비례하여 Z축 이동까지 하게 된다. (다른 오브젝트들 처럼)



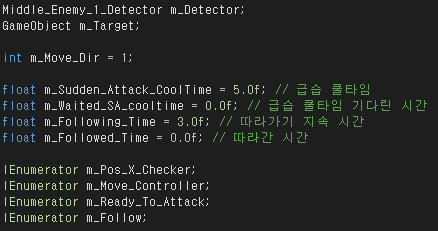
-> 오브젝트에 닿아 소멸 시 초기화 작업 수행 후 풀로 복귀시킨다.

1. **중형 몬스터 AI**

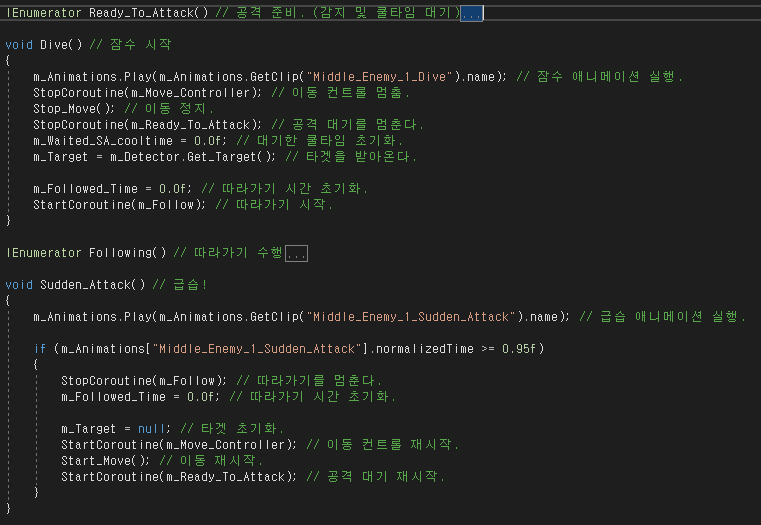


-> 몬스터들은 Enemy 클래스를 상속한다.

Enemy 클래스에는 초기 스탯 설정, 시스템 설정, 이동과 피격, 죽음, 충돌과 같은 메소드들을 다룬다.



-> 중형적이 가지게 되는 변수들과 상태들. 각 상태들은 유일하게 동작한다. Pos\_X 체크와 이동 컨트롤러는 각종 연출에 사용된다.



-> Ready\_To\_Attack()과 Following() 상태.

공격 준비는 플레이어를 감지하기만 하는 상태이다. 감지를 하게 되면 Dive()를 호출하여 Following 상태로 전환 시키고, 플레이어를 따라 Y축과 Z축 이동을 수행하다가 일정 시간이 되면 급습을 수행한다. 급습 연출이 끝나면 위의 X좌표 체크 상태와 이동 컨트롤러를 이용하여 제자리로 복귀한 뒤 다시 공격 준비 상태로 돌아간다.

* **제작 난관 :**

1. 제대로 된 설계를 하지 않고 만들어서 기존 기능들을 계속 갈아엎는 일이 발생.
2. 리소스를 모두 직접 그리다 보니 시간이 많이 소모됨.
3. 애니메이션과 효과음, 배경음을 직접 만들 역량이 안됨. (현재 게임 내에 아무런 사운드도 없으며, 스프라이트 애니메이션 또한 없음)
4. 끝이 없는 게임이다 보니 몬스터와 장애물, 아이템들의 종류와 등장 패턴들이 상당히 많이 필요하지만 1인 개발로서 한계를 느낌.

* **난관 극복 :**

1. 만들고자 하는 기능을 명확하게 하고 필요한 기능들을 정리할 시간을 가짐으로써 해결 (설계 진행). 게임 기획의 중요성을 깨달음.
2. 각 필요 리소스 별로 템플릿 리소스를 만들어 그 위에 덧칠 (스케치) 하는 방향으로 제작 하면서 제작 시간 최소화.
3. 현재 해결 불가. 그림 실력과 작곡 능력을 갖출 필요가 있음.
4. 3번과 유사. 그러나 이는 쉽게 해결할 방법이 존재 (타작품 모방) .견문을 넓힐 필요가 있음.

* **소개 영상 :** <https://www.youtube.com/watch?v=cUWlOkwOv8Y>