**조작법**

주인공 탱크의 조작법은 키보드 w,a,s,d 키를 이용해 이동을 하고 마우스 좌측 버튼을 눌러 마우스 포인트가 가르키는 방향으로 포탄을 발사한다.

본인 목숨을 전부 소모하거나 모든 적을 섬멸할 경우 스페이스바를 이용해서 다시 시작할 수 있다.

**게임진행**

처음에 게임이 실행되면 중앙에 본인의 탱크가 존재하고 맵에 20대의 적 탱크가 스폰 되어있다. 적 탱크 20대를 전부 잡거나 본인의 목숨 4개를 모두 소모하면(목숨이 4개 이므로 처음 탱크까지 5대) 게임 결과가 화면에 표시된다.

**과제에 대한 목표와 가정**

처음에 잡은 계획은 전반적인 시스템이 이미 유니티 튜토리얼만 따라가면 완성할 수 있는 상황이었기 때문에 차이점을 중점으로 개발하기로 하였다.

2인용 게임에서 1인용으로 바뀐만큼 카메라가 기존에는 둘을 모두 잡기위해 움직였지만 주인공 탱크에게 고정하기로 하였고, 기존 탱크 코드를 수정하여 이동과 공격을 캐릭터 범위에 따라 자동적으로 행동을 하게 설정하기로 하였다. 그 외 교수님의 요구사항이었던 맵4배 확장과 적 20기. 목숨4개 등을 하기로 계획을 잡았다.

**씬의 구성**

씬은 단일구성으로 이루어져있다.

카메라, 조명, 이벤트시스템, 메인탱크 와 같이 필수적으로 필요한 것들이 들어있다.

GameManager 에서 게임에 대한 정보와 적 탱크 스폰을 담당한다

CompleteLevelArt 와 Boundaries는 벽, 장애물, 장식물이 들어있는 프리팹이다.

Canvas안에는 남은 적 표시, 남은 목숨 표시, 게임결과 UI를 보관하고 있다.

**자료구조, 알고리즘**

Making폴더 -> Enemy 스크립트

Enemy스크립트에는 적 탱크의 공격과 이동이 담겨있다.

IEnumerator Start() 안에 while (true) yield return new WaitForSeconds(0.5f); 를 넣어 적 위치에 따른 상태 변화와 공격을 0.5초 마다 체크를 하였다.

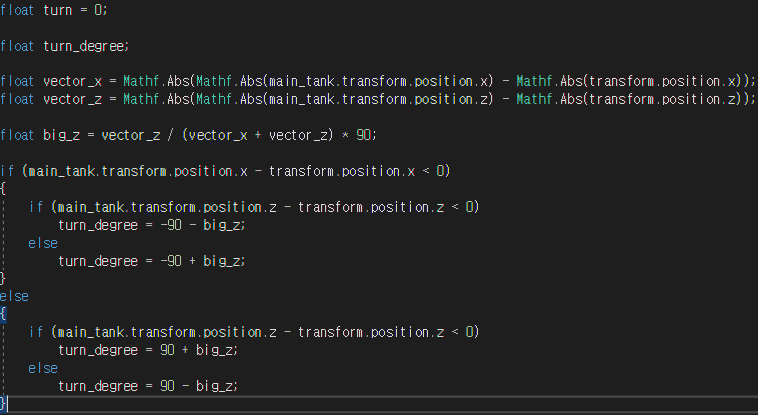
Mathf.Sqrt(Mathf.Pow(main\_tank.transform.position.x - transform.position.x, 2) + Mathf.Pow(main\_tank.transform.position.z - transform.position.z, 2)); 를 이용하여 주인공 탱크와의 거리를 계산해서 20안으로 들어온 경우 모드를 변경하였다.

private void FixedUpdate()에서는 이동과 공격을 담당하는데 평소에는 탱크가 바라보는 방향으로 쭉 가다가

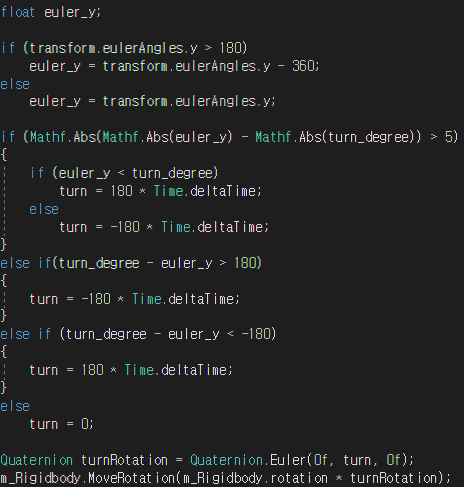
if (Mathf.Abs(transform.position.x - save\_x) + Mathf.Abs(transform.position.z - save\_z) > 1)

통해 0.5초마다 직전 위치와 현재 위치를 비교하여 거리가 1이상 벌어지지 않았으면 충돌하거나 끼인것으로 판단하여 뒤로 가면서 회전을 0.5초 수행한 다음 다시 이동을 한다.

적을 찾은 상태에서는 속도가 1로 줄어버린 다음



주인공 탱크와 자신의 x,z 좌표를 비교한 다음 둘 사이의 각도를 구한다.



그리고 그 각도와 현재 자신의 각도를 비교하여 그 크기가 5도이상 차이가 나게 될 경우 크면 줄이고 작으면 늘린다. 하지만 -180도 다음에는 바로 180도가 되어 위 알고리즘 되로 하면 아래에서 주인공 탱크가 와리가리하면 적 탱크는 반대방향으로 크게 돌아 조준할 경우가 있어 예외처리를 하였다. 그후 차이값 만큼 회전을 시켜주었다.

총알 발사 강도의 경우 현재 주인공 탱크와의 떨어진 거리만큼 강하게 발사를 하게 하였다.

**추가구현 내용**

남은 적과 남은 목숨은 표기가 안되면 알기 어려울거 같아서 유니티에서 기본적으로 제공하는 UI 를 이용하여

**실행 결과(실행 화면)**

