**Recoil Machanism 개요**

**GunController.cs**

TryFire():

public void TryFire()

{

//OwnerPlayer가 아니거나 장전중이라면 return

if (!isOwner || isReloading)

return;

//Fire버튼이 눌리고 있는 상태이며 Fire 쿨타임중이 아닐때

if (isShotButtonActive && isOnShotCooltime == false)

{

//장비중인 Gun의 장탄수가 0보다 많을때

if (equippedGun.inMagazine > 0)

{

//발사 방향 벡터 계산

Vector3 targetPos = GetShotDirection();

//발사 방향 벡터를 인자로 Fire 메시지 발신

SendToMessage(targetPos);

//발사 쿨타임 활성화

StartCoroutine("StartShotCooltime");

//발사 후 반동지수가 최대가 아닐경우 반동지수 증가

if (currRecoilRadius < maxRecoilRadius)

currRecoilRadius += Time.deltaTime;

}

// 장탄수가 0일경우

else

{

//장전 수행

SendToMessage(KeyEventCode.RELOAD);

//tryReload();

}

}

}

SearchLight는 P2P 방식에 유사한 멀티플레이어 게임이기 때문에 대부분의 게임 로직이 메시지 발신 및 수신 과정을 거치게 되며 GunController에서 TryFire() 즉 총기 발사 요청 동작이 수행될 때마다 GetShotDirection()을 통해 현재 총기 반동 지수 (currRecoilRadius, 정확하게는 현재 총기 반동 범위)에 따른 발사 방향을 계산하여 해당 방향 벡터를 기반으로 Fire 메시지를 발신하고 해당 Fire 메시지 수신부에서 다시 GunController의 FireAction()을 호출하여 실제 발사를 수행한다.

발사 메시지 발신 후에는 현재 총기 반동 지수 (currRecoilRadius)를 증가시키는 방식으로 총기가 연속으로 발사될 때마다 그 반동이 높아져 명중률이 저하되는 효과를 구현했다.

GetShotDirection():

public Vector3 GetShotDirection()

{

//사격 목표 위치

Vector3 circlePos = transform.position + transform.forward \* shotDistance;

//원의 둘레 위 랜덤 좌표 (currRecoilRadius는 반동 지수)

Vector2 randomCirclePoint = UnityEngine.Random.insideUnitCircle.normalized \* currRecoilRadius;

Vector3 targetPos = new Vector3(circlePos.x + randomCirclePoint.x, 0, circlePos.z + randomCirclePoint.y);

return targetPos;

}

circlePos: 플레이어의 현재 위치 + 사정거리

randomCirclePoint: 원의 둘레 위 랜덤 좌표

유니티의 Random.insideUnitCircle은 반지름 1의 원에 대한 내부 랜덤 좌표를 반환하기 때문에 이를 normalized 하면 magnitude가 1인 벡터, 즉 원의 둘레 위에 있는 랜덤 좌표를 얻을 수 있다.

해당 좌표의 x값과 y값을 3차원 벡터 값으로 변환하여 정확한 3차원 사격 목표 좌표를 얻을 수 있다.

TryFire에서 발사 메시지가 발신될 때마다 반동 수치 (반동 범위)에 해당하는 currRecoilRadius가 증가하기 때문에 발사시마다 반동 Circle이 커지기 때문에 그 랜덤좌표의 간격도 점점 넓어져 결과적으로 명중률의 저하를 나타내게 된다.