**Detect 개요**

**Detect.cs**

게임의 핵심 시스템이 되는 시야 시스템을 담당하는 스크립트, 게임에서는 상대 플레이어가 시야각에 들어오지 않으면 아예 보이지 않게되며 시야각에 들어왔을 때만 상대 플레이어가 보이는 식으로 작동한다.

OnTriggerStay()

감지된 플레이어 렌더러 관리

private void OnTriggerStay(Collider other)

{

//other가 Player면서 현재 detect의 ownerPlayer가 아닐때 (other가 감지 범위 내에 있을때)

if (other.tag == "Player" && other.gameObject != transform.parent.gameObject)

{

//detectedPlayers 딕셔너리에 해당 플레이어 오브젝트의 이름이 있을때

if(detectedPlayers.ContainsKey(other.name))

{

bool isInSight = CheckIsTargetInSight(other.transform.position);

// other가 시야각 내에 위치했으면서 other의 렌더러 flag가 false일때

if (isInSight == true && detectedPlayersFlag[other.name] == false)

{

//해당 other의 렌더러를 활성화 시킨다.

detectedPlayers[other.name].setRendererActive(true);

detectedPlayersFlag[other.name] = true;

Debug.Log("InSight!");

}

// other가 시야각 바깥에 있으면서 other의 렌더러 flag 가 true일때

else if(isInSight == false && detectedPlayersFlag[other.name] == true)

{

//해당 other의 렌더러를 비활성화 시킨다.

detectedPlayers[other.name].setRendererActive(false);

detectedPlayersFlag[other.name] = false;

Debug.Log("OutSight");

}

}

else

{

//detectedPlayers에 해당 플레이어가 등록되어있지 않으므로 등록

detectedPlayers.Add(other.name, other.GetComponent<Player>());

detectedPlayersFlag[other.name] = false;

Debug.Log(string.Format("{0} added in detectedPlayers dictionary", other.name));

}

}

}

SphereCollider로 감지 범위를 구현하고 해당 감지 범위내에서 다른 Player가 Trigger되면 detectedPlayers 딕셔너리에 해당 플레이어를 등록한다.

TriggerStay 즉 다른 Player가 감지 범위 내에 위치했을 시 프레임마다 해당 Player가 OwnerPlayer의 시야각 내에 위치했는지를 체크하여 감지범위 내의 Player들의 렌더러를 컨트롤한다.

플레이 중에 OwnerPlayer의 방향이 빈번하게 바뀌는 만큼 높은 빈도로 상대 Player의 정보를 읽어와야 하는 상황을 방지하기 위해 딕셔너리를 활용하여 Trigger되는 Player의 오브젝트 이름을 통해 한번 감지된 후 아직 TriggerExit되지 않은 Player의 정보를 빠르게 가져올 수 있다.

CheckIsTargetInSight():

bool CheckIsTargetInSight(Vector3 otherPos)

{

//플레이어 전방 벡터

Vector3 playerForward = transform.forward;

//플레이어에서 목표를 가리키는 벡터

Vector3 playerToTarget = otherPos - transform.position;

//두 벡터의 내적을 이용한 두 벡터의 사이각 산출

float angleToTarget = Mathf.Acos(Vector3.Dot(playerForward, playerToTarget) / (playerForward.magnitude \* playerToTarget.magnitude)) \* Mathf.Rad2Deg;

//플레이어의 시야각 angle의 절반보다 두 벡터의 사이각이 작다면 시야각 내에 위치한것

if (angle / 2 > angleToTarget)

{

return true;

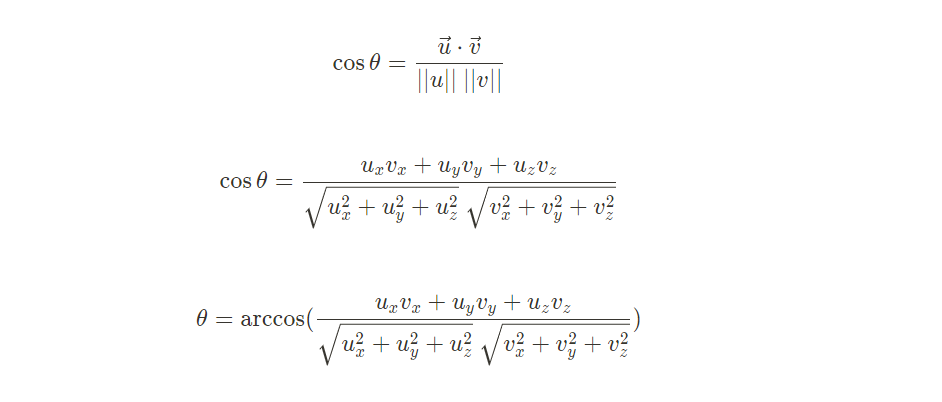
}

return false;

}

trigger에서 감지된 다른 player의 위치를 기반으로 플레이어의 전방 벡터와 플레이어 -> 상대 플레이어 벡터를 활용하여 두 벡터의 사이각을 통해 시야각에 위치했는지 확인할 수 있다.

※ 두 벡터의 사이각을 구하는 공식



OnTriggerExit()

감지된 플레이어의 정보 삭제

private void OnTriggerExit(Collider other)

{

if (other.tag == "Player")

{

//detectedPlayers에 등록되어있지 않은 플레이어가 TriggerExit 되었을 경우 에러 로그 출력

if (detectedPlayers.ContainsKey(other.name) == false)

{

Debug.LogError("Unaddeded Player exit in detectedPlayers");

}

else

{

//TriggerExit된 Player의 정보 삭제

detectedPlayers[other.name].setRendererActive(false);

detectedPlayers.Remove(other.name);

detectedPlayersFlag.Remove(other.name);

}

}

}

다른 Player가 TriggerExit 될 경우 해당 detectedPlayers 딕셔너리에서 해당 플레이어의 정보를 삭제한다.