[서버 내 캐릭터 동기화 관련 클래스]

1. **PhotonManager_클래스 :** 포톤 서버에 접속해 플레이어가 자신의 캐릭터를 생성하도록 돕고, 게임 내 진행을 책임진다. 맵에 한 개만 존재해야 함.

[코드설명]

```
void Start()
{
    PhotonNetwork.ConnectUsingSettings("1.0"); // 버전 설정 및 로비 진입
    if(GAME == true) // 게임 진행되는 방인지 체크
    {
        newchr = true; // 전투가 가능한 캐릭터를 생성하도록 진행.

        mo = false; // mo = Make_Object : 캐릭터 생성 여부를 체크함.


    }
}
참조 0개
void OnJoinedLobby()
{
    PhotonNetwork.JoinOrCreateRoom("Room", new RoomOptions() { MaxPlayers = 6 }, TypedLobby.Default); // 로비에 진입하면 룸에 들어가거나 새로 룸
}
참조 0개
void OnJoinedRoom()
{
    if (GAME == false)
    {
        if (lobbyCamera)
        {
            // lobbyCamera = 게임 맵마다 존재하는 카메라 : 전투 시작전 맵 전체를 비춤.
            // 전투가 없는 대기실에선 해당 카메라를 꺼둠.
            lobbyCamera.SetActive(false);
        }
    }
}
```

- 해당 클래스는 자동으로 시작하자마자 서버(로비)에 접속해 룸에 들어가도록 작성됨.
- 또한, 캐릭터 생성 여부를 FALSE로 체크하고, 캐릭터 생성을 진행하도록 함.
- mo 변수는 자신의 캐릭터 생성 후 true로 바뀜.
- OnJoinedLobby(로비 접속 시), OnJoinedRoom(룸 접속 시)은 Photon.MonoBehaviour를 상속한 PhotonManager 클래스에서 별도 호출 없이 자동 호출된다.
    - ■ 캐릭터는 **GAME 변수**를 통해 대기 룸인지 게임이 진행되는 곳인지 체크해 캐릭터의 전투 가능 여부를 결정한다.
- **lobbyCamera 변수**는 게임이 진행되는 레벨에서 게임 맵 전체를 보여주는 카메라.

```
void Update()
{
    if (go != null && Hide == true) // go : Player 관련 제어 클래스 객체, Hide : 해당 맵이 숨바꼭질 맵인지 체크
    {
        GameObject[] box = GameObject.FindGameObjectsWithTag("KillBox"); // 숨바꼭질 맵에선 KillBox를 전부 먹어야 방장을 쓰러뜨릴 수 있음.
        boxnum = box.Length;
        if (go.pla.p1 == true && go.f.health >= 10 && boxnum != 0) // go.pla은 몇번째 플레이어인지 표시하는 객체. go.pla.p1이 true이면 방장이고, 방장은 KillBox가 있는 동안 무적 (go.pla.p1~p6까지 존재)
        {
            go.f.health = 2000;
            go.f.plusdmage = 9999999; // 방장의 데미지 또한 최대치로 증가
        }
        else if (go.pla.p1 == true && go.f.health >= 10 && boxnum == 0)
        {
            go.f.health = 0;
        }

        else
        {
            go.f.sm = 0; // 방장의 필살기 게이지 = 0
        }
    }

    if(go && GAME == true)
    {
        Destroy(go.nicknameAP);
    }
    if(UserData.roomlock == false && gameserv == false) // roomlock은 게임 시작 전에 들어와 있던 플레이어만 true이고, gameserv의 같은 대기룸이 아닌 게임 맵에서만 true.
    {
        UserData.notice = true; // 경고 메시지를 띄우고, 게임 맵에 들어오지 못하게 막고, 다시 메뉴가 있는 곳으로 튀겨낸다.
        SceneManager.LoadScene(2, LoadSceneMode.Single);
    }
    else
        if (newchr == true) // 플레이어 캐릭터가 생성될 때 플레이어가 들어온 순서에 따라 순번을 매김.
        {
            if (user == 1) // user는 현재 플레이어의 수. player1~6은 각 플레이어를 대표함. 플레이어 데이터를 순서에 따라 다른 번수에 저장함.
            {
                player1 = NOW; // NOW는 자신의 플레이어 데이터를 나타내는 클래스 객체

                pl(); // 플레이어 생성 함수(NOW에 들어있는 플레이어 데이터에 따라 다른 캐릭터를 생성한다.)
            }
            else if (user == 2)
            {
                player2 = NOW;
                pl();
            }
            else if (user == 3)
            {
                player3 = NOW;
                pl();
            }
            else if (user == 4)
            {
                player4 = NOW;
                pl();
            }
            else if (user == 5)
            {
                player5 = NOW;
                pl();
            }
            else if (user == 6)
            {
                player6 = NOW;
                pl();
            }
        }
```

- Update 에서 플레이어 생성과 관련된 작업이 실행된다.

  ⇨ 플레이어가 생성되지 않을 경우 고려해 반복적으로 결과 체크해 실행하도록 하기 위함.

- 게임 내 모드가 **숨바꼭질 모드와 일반 전투 모드**로 나뉘는데 이를 고려해 숨바꼭질 모드일 경우에만 방장을 무적으로 설정

- 현재 유저의 수에 따라 현재 플레이어의 순번을 다르게 설정함.

  ⇨ 여기서 정한 순번은 다른 플레이어가 나가지 않는 이상 바뀌지 않음.

- NOW 는 플레이어의 정보를 저장한 객체로 해당 객체에 있는 데이터로 서버에 캐릭터를 생성한다.

  ⇨ pl 함수에 해당 내용이 들어간다. (PhotonNetwork.Instantiate 함수 이용)

2. **PhotonCheck 클래스 : 플레이어를 실시간으로 체크하고, 이를 게임에 반영해 문제가 발생하지 않도록 한다.**

[코드 설명]

```
IEnumerator Userspace() // PhotonManger 객체에서 유저 수를 다시 체크하도록 함. 또한, 새로운 유저의 데이터를 동기화.
{
    if (pm.user != PhotonNetwork.countOfPlayers)
    {
        pm.user = PhotonNetwork.countOfPlayers;
    }
    yield return new WaitForSeconds(2); // 2초 정도 텀을 줌. => 플레이어가 들어오고 약간 지나서 데이터 체크하도록 함.

    pm.newchr = true; // 플레이어 데이터 불러오기

}
// Update is called once per frame
Unity 메시지 | 참조 0개
void Update()
{
        if (pm)
        {
            StartCoroutine(Userspace()); // 실시간으로 유저 데이터 동기화

            // 아래는 플레이어 닉네임 화면 내 동기화 과정
            if (pm.ST == true)
            {
                pm.playername1 = pm.go.t1;
                pm.playername2 = pm.go.t2;
                pm.playername3 = pm.go.t3;
                pm.playername4 = pm.go.t4;
                pm.playername5 = pm.go.t5;
                pm.playername6 = pm.go.t6;
            }
            if (pm.go)
            {
                if (pm.go.pla.pl == true)
                {
                    pm.playername1.text = pm.go.NICK;
                    pm.playername2.text = null;
                    pm.playername3.text = null;
                    pm.playername4.text = null;
                    pm.playername5.text = null;
                    pm.playername6.text = null;
                }
```

- Update 에서 Userspace 를 이용해 유저 수를 지속적으로 갱신하도록 함.
- 유저 수를 갱신한 후에 실시간으로 화면 내 유저의 닉네임 표시 부분도 갱신함.
- pm 은 PhotonManager 객체를 의미하며 newchr 의 값을 true 로 바꾸는 이유는

  새로운 유저 데이터를 등록하기 위해서이다. 유저 데이터를 등록하는 이유는 캐릭터의 스텟 및 착용 아이템 등을 서버에서 동기화하기 위함.