

Lobby ve Oda İşlemleri

1)Genel anlatım | Login Form işlemleri | Oda kurma işlemleri

İlk olarak GameManager dosyası oluşturuyoruz. Panel vs. hazırladıktan sonra (Buraları anlatmıyorum) Awake fonksiyonunda otomatik senkronizasyonu açıyoruz.

```
0 references
public void Awake()
{
    /*Otomatik sahnenin senkronizasyon olmasını bize sağlar.*/
    PhotonNetwork.AutomaticallySyncScene = true;
}
```

1- Bağlantı bölümü



1.İsim alabildiğimiz taktirde bağlantıyı kuruyoruz.

```
/*Gelen kullanıcı ismini burada alacağız.*/
0 references
public void OnLoginButtonClicked()
{
    string playerName = OyuncuİsimInput.text;
    if(!playerName.Equals("")){
        PhotonNetwork.LocalPlayer.NickName = playerName;
        PhotonNetwork.ConnectUsingSettings();
    }
    else{
        Debug.Log("Oyuncu ismi gelmedi");
    }
}
```

2.Bağlantı başarılı olduğunda çalışan fonksiyonumuz.

```
5 references
public override void OnConnectedToMaster()
{
    /*Bağlantı başarılı olduğu anda oto çalışan fonksiyondur.
    Burada artık seçim panelini çıkartıyoruz.*/
    SetActivePanel(SecimPanel.name);
}
```

2-Seçim paneli (Oda Kurma)



→ Oda Kur buton →



1.Oda kurma fonksiyonumuz

```
0 references
public void OnCreateRoomButtonClicked()
{
    string roomName = OdaAdiInput.text;
    roomName = (roomName.Equals(string.Empty)) ? "Oda " + Random.Range(100, 100000) : roomName;
    //MaxPlayer değeri türü byte olduğunu unutma
    byte.TryParse(MaksimumOyuncuInput.text, out byte MaxPlayer);
    MaxPlayer = (byte) Mathf.Clamp(MaxPlayer, 2, 8);

    RoomOptions options = new RoomOptions{
        MaxPlayers = MaxPlayer
    };
    PhotonNetwork.CreateRoom(roomName, options, null);
}
```

2)Oda listesi altyapı işlemleri | Cache ve Kayıtların alt yapısı

Odamızı bir üstte kurduk. Şimdi ise, kurulmuş olan odaların listesini görme işlemi gerçekleştirmemiz gerekiyor.

1-Kurulan odaları listeden alma

```
//Odaları tutacağımız liste
0 references
private Dictionary<string, RoomInfo> OdaCacheList;

//O an kaç odam varsa onu liste halinde tutabilmemizi sağlayacak
0 references
private void UpdateCachedRoomList(List<RoomInfo> roomList)
{
    //RoomInfo: Odaların özelliklerini durumlarını almamızı sağlar
    foreach (RoomInfo info in roomList)
    {
        //Kullanıcı kapasitesi dolu olan odaları göstermemizi sağlayan methodu yazacağız.
        if(!info.IsOpen || !info.IsVisible || !info.RemovedFromList){
            //Bu listenin güncelliğini sağlamamız gerekiyor. Aşağıda işi biten listeleri kaldırıyoruz.
            if(OdaCacheList.ContainsKey(info.Name)){
                OdaCacheList.Remove(info.Name);
            }
            /*Eğer kaldırma işlemi yapılırsa continue ile tekrardan
            en baştan foreach döngüsünü çalıştırıyoruz*/
            continue;
        }
        /*Eğer daha önceden bu isimde bir oda varsa yeni oda bilgileri ile
        aslında bir update güncelleme işlemi yapılıyor denebilir.*/
        if(OdaCacheList.ContainsKey(info.Name)){
            OdaCacheList[info.Name] = info;
        }
        else{
            //Daha önce bu listeye hiçbir zaman ekleme yapılmadığını belirtiyor.
            OdaCacheList.Add(info.Name, info);
        }
    }
}
```

2-Listedeki elemanları sıralama

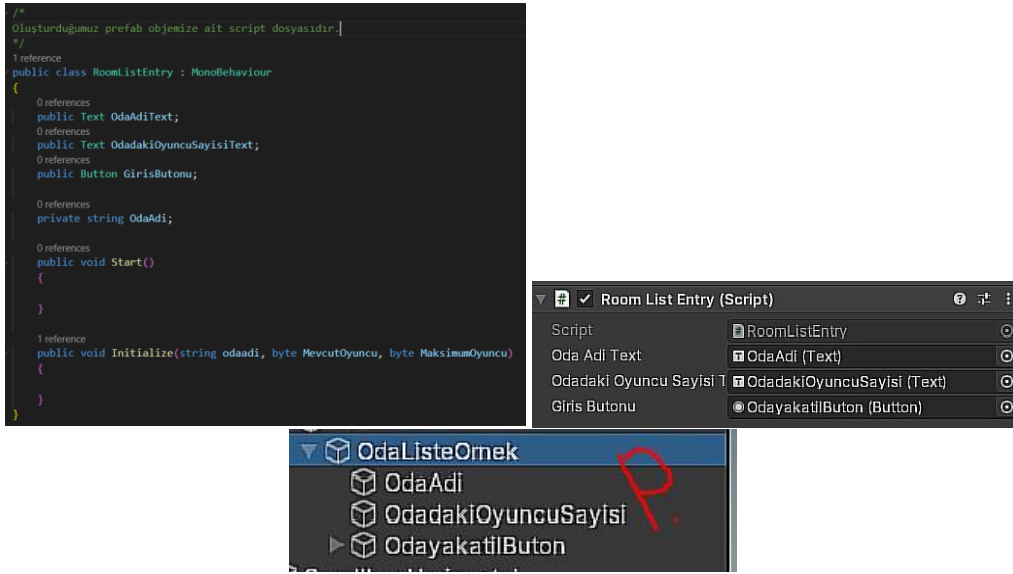
```
1 reference
public GameObject OdalisteriSatinPrefab;

private Dictionary<string, RoomInfo> OdaCacheList;
1 reference
private Dictionary<string, GameObject> OdalisteriElemanlari;

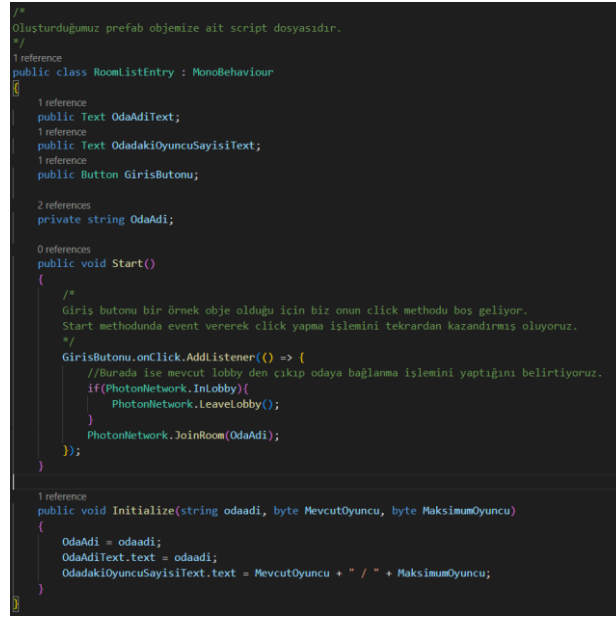
// Liste halindeki odaları sıralamamızı sağlar.
0 references
private void UpdateRoomListView()
{
    //Oluşturduğumuz güncel listeleri kullanarak odaları satır satır yazmamızı sağlar.
    foreach (RoomInfo info in OdaCacheList.Values)
    {
        /*Burada ne kadar satır var ise o kadar liste oluşturuyoruz ve
        aldığımız bilgiler ile o objenin içeriğini dolduruyoruz. */

        /*Photon instantiate komutunu kullanmamamızın sebebi bizim
        zaten listeyi photon güncel olarak vermesinden kaynaklı yani
        tekrardan bizim sunucu tarafı liste oluşturmamıza gerek yok
        */
        GameObject entry = Instantiate(OdalisteriSatinPrefab);
        //Oluşturduğumuz entry odalisteriContent içerisinde çocuk obje olarak atıyorum
        entry.transform.SetParent(OdalisteriContent.transform);
        //Scale değerinde herhangi bir kayıp olmasını istemiyoruz.
        entry.transform.localScale = Vector3.one;
        /*Oluşturduğumuz oda listesi elemanı da kendisine ait hazırladığımız
        listenin içerisine atıyoruz */
        OdalisteriElemanlari.Add(info.Name, entry);
    }
}
```

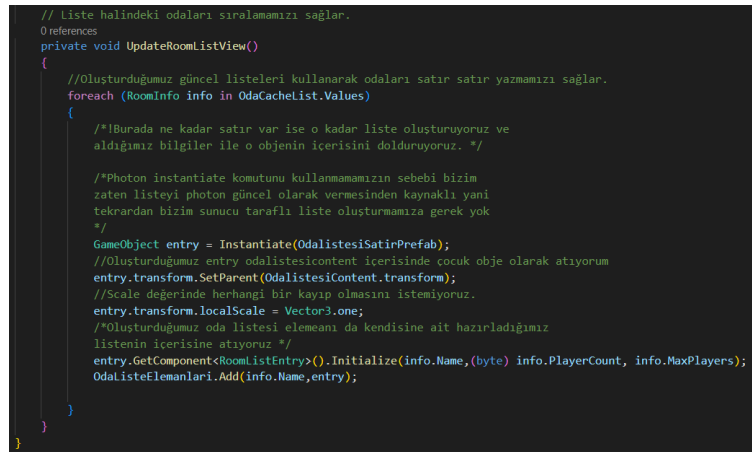
1. Prefab oluşturma



2. Prefab objesinin içerisindeki elemanları doldurma.



3. Prefab objesine ait gerekli verileri gönderme



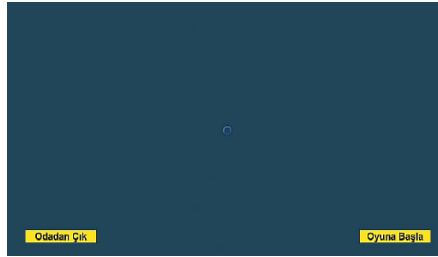
3)Join işlemlerin yapılması | Oyuncu satır objesinin oluşturulması

1-Şimdiki yapacağımız işlem ise mevcut odalar içerisinde herhangi bir değişiklik olursa yani bir güncelleme olursa biz bunu bir callback fonksiyonu ile elde edebiliriz.

```
/*Odalara yeni bir oyuncu girdiğinde, yeni bir oda kurulduğunda ya da odalar ile ilgili güncelleme yapıldığında bu fonksiyon çalışacaktır. */  
5 references  
public override void OnRoomListUpdate(List<RoomInfo> roomList)  
{  
    UpdateCachedRoomList(roomList);  
    UpdateRoomListView();  
}
```

2-Oyuncumuzun odaya giriş callback fonksiyonunu hazırlama

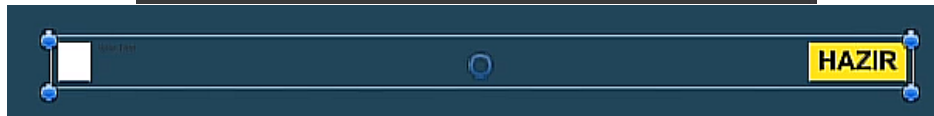
```
0 references  
public GameObject OyunculistesisiSatirPrefab;  
  
0 references  
private Dictionary<int, GameObject> OyuncuListeElemanlari;
```



1.Odadaki oyuncular için prefab oluşturma

```
4 references  
public class PlayerListEntry : MonoBehaviour  
{  
    1 reference  
    public Text OyuncuAdi;  
    0 references  
    public Button OyuncuHazirbuton;  
    0 references  
    public Image OyuncuSprite;  
    1 reference  
    private int oyuncuid;  
    0 references  
    private bool Hazirmi;  
  
    0 references  
    public void Start()  
    {  
    }  
  
    2 references  
    public void Initialize(int playerId, string playerName)  
    {  
        oyuncuid = playerId;  
        OyuncuAdi.text = playerName;  
    }  
  
    2 references  
    public void SetPlayerReady(bool playerReady)  
    {  
    }  
}
```

#	Player List Entry (Script)
Script	PlayerListEntry
Oyuncu Adı	OyuncuAd (Text)
Oyuncu Hazirbuton	OyuncuHazirbuton (Button)
Oyuncu Sprite	OyuncuIcon (Image)



4)Join işlemlerin yapılması | Oyuncu HAZIRIM butonunun yapımı

Prefab objemizi bir önceki bölümde oluşturduk. Şimdiki yapacağımız işlem ise hem prefab dosya içerisini doldurma işlemi yapacağız. Hem de oyuncu listesini getirme ve sıralama işlemlerini gerçekleştireceğiz.

1-Prefab objemizin içeriğini doldurma

```
6 references
public class PlayerListEntry : MonoBehaviour
{
    1 reference
    public Text OyuncuAdi;
    3 references
    public Button OyuncuHazirbuton;
    1 reference
    public Image OyuncuSprite;
    2 references
    private int oyuncuid;
    5 references
    private bool Hazirmi;
}

0 references
public void Start()
{
    /*Actor number oyuncunun id barındırır. Eğer bağlantı kopar
    oyuncu çıkar, sistemden düşme gibi durumları kontrol ediyoruz.
    Bu durumlarda hazırım butonunu kapatıyoruz.*/
    if(PhotonNetwork.LocalPlayer.ActorNumber != oyuncuid){
        OyuncuHazirbuton.gameObject.SetActive(false);
    }
    else{
        //Eğer herhangi bir sorun yok ise buradan devam edeceğiz.
        //Oyuncunun hazır olup olmadığını bu özellik ile kontrol ediyoruz.
        Hashtable props = new Hashtable{
            ("IsPlayerReady",Hazirmi)
        };
        PhotonNetwork.LocalPlayer.SetCustomProperties(props);
        OyuncuHazirbuton.onClick.AddListener(() => {
            //True ise false - false ise true yapar
            Hazirmi = !Hazirmi;
            SetPlayerReady(Hazirmi);
            //Aynı isimle özellik tanımlarsak onun üzerine yazacak.
            Hashtable props2 = new Hashtable{
                ("IsPlayerReady",Hazirmi)
            };
            PhotonNetwork.LocalPlayer.SetCustomProperties(props2);

            /*Bu fonksiyon sadece oda kurucusu ilgilendirir.*/
            if(PhotonNetwork.IsMasterClient){
                FindObjectOfType<GameManager>().LocalPlayerPropertiesUpdated();
            }
        });
    }
}

3 references
public void Initialize(int playerId, string playerName)
{
    oyuncuid = playerId;
    OyuncuAdi.text = playerName;
}

4 references
public void SetPlayerReady(bool playerReady)
{
    //Ternary
    OyuncuHazirbuton.GetComponentInChildren<Text>().text = playerReady ? "OK !" : "Hazırım";
    OyuncuSprite.enabled = playerReady;
}
}
```

2-Prefab obje elemanlarını getirme ve listeleme

```
/*Herhangi bir odaya girdiğimizde tetiklenecek olan callback fonksiyonu*/
5 references
public override void OnJoinedRoom()
{
    /*Artık odaya girdiğimiz için buradaki listeye ihtiyacımız kalmadı.
    İhtiyacımız olmadığı için sildiğimizde ön bellekte kapladığı yeri
    temizlemiş olacağız. Perf. kazancı sağlıyoruz.*/
    OdaCacheList.Clear();
    SetActivePanel(OdaiciPanel.name);

    /*Odanın içerisindeki elemanları da tutacağımız listeyi hazırlıyoruz.*/
    if(OyuncuListeElemanlari == null){
        OyuncuListeElemanlari = new Dictionary<int, GameObject>();
    }
    foreach (Player p in PhotonNetwork.PlayerList)
    {
        GameObject entry = Instantiate(OyuncuListesiSatirPrefab);
        entry.transform.SetParent(OdaiciPanel.transform);
        entry.transform.localScale = Vector3.one;
        entry.GetComponent<PlayerListEntry>().Initialize(p.ActorNumber,p.NickName);
        //Hazırım butonu entegrasyonu yaptık
        if(p.CustomProperties.TryGetValue("IsPlayerReady", out object IsPlayerReady)){
            entry.GetComponent<PlayerListEntry>().SetPlayerReady((bool) IsPlayerReady);
        }
        OdaListeElemanlari.Add(p.UserId,entry);
    }
}
}
```

3-Oyunu Başlat fonksiyonu

```
/*Herhangi bir odaya girdiğimizde tetiklenecek olan callback fonksiyonu*/
5 references
public override void OnJoinedRoom()
{
    /*Artık odaya girdiğimiz için buradaki listeye ihtiyacımız kalmadı.
    İhtiyacımız olmadığı için sildiğimizde ön bellekte kapladığı yeri
    temizlemiş olacağız. Perf. kazancı sağlıyoruz.*/
    OdaCacheList.Clear();
    SetActivePanel(OdaiciPanel.name);

    /*Odanın içerisindeki elemanları da tutacağımız listeyi hazırlıyoruz.*/
    if(OyuncuListeElemanlari == null){
        OyuncuListeElemanlari = new Dictionary<int, GameObject>();
    }
    foreach (Player p in PhotonNetwork.PlayerList)
    {
        GameObject entry = Instantiate(OyuncuListesiSatirPrefab);
        entry.transform.SetParent(OdaiciPanel.transform);
        entry.transform.localScale = Vector3.one;
        entry.GetComponent<PlayerListEntry>().Initialize(p.ActorNumber,p.NickName);
        //Hazırım butonu entegrasyonu yaptık
        if(p.CustomProperties.TryGetValue("IsPlayerReady", out object IsPlayerReady)){
            entry.GetComponent<PlayerListEntry>().SetPlayerReady((bool) IsPlayerReady);
        }
        OdaListeElemanlari.Add(p.UserId,entry);
    }
    //Oyunu başlatan kişi odayı kuran kişidir.
    /*Bool döndüren bir fonksiyon çalıştırıyoruz. Bunun nedenli ise odaya başlama işinde kimin
    karar vereceğini belirttiğimiz fonk.*/
    OyunaBaslaButon.gameObject.SetActive(CheckPlayersReady());

    //Oyuncuların sahneye gelip gelmediğini öğreneceğimiz fonk. İleride kullanacağız.
    Hashtable props = new Hashtable{
        {"PlayerLoadedLevel",false}
    };
    PhotonNetwork.LocalPlayer.SetCustomProperties(props);
}

1 reference
private bool CheckPlayersReady()
{
    if(!PhotonNetwork.IsMasterClient){
        return false;
    }
    //Oyunculara toplu şekilde erişme
    foreach (Player p in PhotonNetwork.PlayerList)
    {
        if(p.CustomProperties.TryGetValue("IsPlayerReady", out object IsPlayerReady)){
            if(!((bool) IsPlayerReady))
                return false;
        }
        else{
            return false;
        }
    }
    return true;
}
```

5)Oda kurma ve test | Oda listesi test

1-Listeleri hata almamak için örneklememiz gerekiyor.

```
0 references
public void Awake()
{
    /*Otomatik sahnenin sekranizasyon olmasını bize sağlar.*/
    PhotonNetwork.AutomaticallySyncScene = true;

    /*Bu liste işlemlerinde hata almamak için awake fonk.
    new diyerek örneklememizi yapıyoruz.*/
    OdaCacheList = new Dictionary<string, RoomInfo>();
    OdaListeElemanlari = new Dictionary<string, GameObject>();
}


```

2-Oda listesi butonuna tıklandığında oda listesini görme

```
/*Lobby de olup olmadığımızı kontrol edip butona tıklandığında
Liste panelinin açılmasını istedik.*/
0 references
public void OnRoomListButtonClicked()
{
    if(!PhotonNetwork.InLobby){
        PhotonNetwork.JoinLobby();
    }
    SetActivePanel(OdaListePanel.name);
}


```

6)Sistem fonksiyonlarının yazılması

Sıradaki işlemimiz ise kodlarımızı senkron yapacağız.

1-OdaListeElemanları'nı temizliyoruz.

```
/*Bu fonksiyon oda değişebilir, oyuncu listeden çıkabilir, oyuncu lobbyden çıkabilir, oyuncu oyundan çıkıp tekrar listeye erişmek isteyebilir vs. bu durumları yakalayabilmek için bir oda liste elemanlarını tutan listemizi temizlememiz gerekiyor. Güncel listeyi yakalamamız gerekiyor.*/
5 references
private void ClearRoomListView()
{
    /*Eğer Foreach döngüsünü ilk olarak yaparsak, objeler olduğu gibi sistemde kalır. İlk olarak objeleri sonra listeyi temizliyoruz.*/
    foreach(GameObject entry in OdaListeElemanlari.Values){
        Destroy(entry.gameObject);
    }
    /*Yeni elemanların eklenebilmesi ve arda kalan veri ve obje olmaması açısından*/
    OdaListeElemanlari.Clear();
}
```

2-Yukarıdaki ClearRoomListView fonksiyonunu çağırıyoruz.

```
/*Odalara yeni bir oyuncu girdiğinde, yeni bir oda kurulduğunda ya da odalar ile ilgili güncelleme yapıldığında bu fonksiyon çalışacaktır. */
5 references
public override void OnRoomListUpdate(List<RoomInfo> roomList)
{
    ClearRoomListView();
    UpdateCachedRoomList(roomList);
    UpdateRoomListView();
}
```

3- Oyuna başla butonunu da hazırlıyoruz.

```
/*Local nevot oyuncunun, özelliklerini update edip PlayerListUpdate script dosyasında çağırıldıkça, bu fonksiyon da oluşturulması gerekiyor.*/
5 references
public void LocalPlayerPropertiesUpdated()
{
    //Odayı kurmaya çalışacak olan baslat butonu
    OyunBaslaButon.gameObject.SetActive(CheckPlayersReady());
}
```

4-Diğer kalan fonksiyonlarımızı da hazırlıyoruz.

1.Lobby giriş çıkışlarında liste temizleme

```
4 references
public override void OnJoinedLobby()
{
    /*Eğer lobby girerse bütün değerleri tazelem için listeleri temizliyoruz.*/
    OdaCacheList.Clear();
    ClearRoomListView();
}

3 references
public override void OnLeftLobby()
{
    /*Eğer lobby çıkarsa bütün değerleri tazelem için listeleri temizliyoruz.*/
    OdaCacheList.Clear();
    ClearRoomListView();
}
```

2.Hata alma durumlarındaki fonksiyonları hazırlıyoruz.

```
3 references
public override void OnCreateRoomFailed(short returnCode, string message)
{
    /*Odayı kurmaya çalışıp kuramazsak, tekrardan ana sayfaya dönüyoruz.*/
    SetActivePanel(SecimPanel.name);
}

2 references
public override void OnJoinRoomFailed(short returnCode, string message)
{
    /*Odaya girmeye çalışıp odaya giriş yapamazsa, tekrardan ana sayfaya dönüyoruz.*/
    SetActivePanel(SecimPanel.name);
}

2 references
public override void OnJoinRandomFailed(short returnCode, string message)
{
    /*Odaya random girmeye çalışıp odaya giriş yapamazsa, sistemsel hata da olabilir ya da açık oda olmayada bilir. Bu kontrollere bakarak işlem yapmamız gerekiyor.*/

    /*Eğer random oyun butonuna basarsa ve random oda yok ise otomatik olarak 2 kişilik oda kurup oyuncuyu gönderiyoruz.*/
    SetActivePanel(SecimPanel.name);
    string roomName = OdaAdInput.text;
    roomName = (roomName.Equals(string.Empty)) ? "Oda " + Random.Range(100, 100000) : roomName;
    RoomOptions options = new RoomOptions(MaxPlayers = 2);
    PhotonNetwork.CreateRoom(roomName,options,null);
}
```

3.Odadan çıktığımdaki işlemler

```
5 references
public override void OnLeftRoom()
{
    /*Secim paneline gönderiyoruz.*/
    SetActivePanel(SecimPanel.name);
    /*Eğer odadan çıkarsak birinci olarak kesinlikle oyunculara ait liste elemanlarını temizlememiz gerekiyor.*/
    foreach(GameObject entry in OyuncuListeElemanlari.Values){
        Destroy(entry.gameObject);
    }
    OyuncuListeElemanlari.Clear();
    OyuncuListeElemanlari = null;
}
```

4.Yeni giren oyuncu olursa

```
2 references
public override void OnPlayerEnteredRoom(Player newPlayer)
{
    /*Eğer ki odaya yeni bir oyuncu girse, oda liste elemanlarına ve satırına bu kullanıcıyı eklemem gerekiyor.*/
    /*Eğer bu işlemi yapmazsa oyuncular bağlantı yaptığı birbirlere göremlerini göremeyebilir. Ya da görmez!*/
    GameObject entry = Instantiate(OyuncuListeSatiirPrefab);
    entry.transform.SetParent(OdaiciPanel.transform);
    entry.transform.localScale = Vector3.one;
    entry.GetComponent<PlayerListEntry>().Initialize(newPlayer.ActorNumber,newPlayer.Nickname);
    /*Burada hazırım butonu entegrasyonu yapmamızın sebebi zaten oluşturma işlemini yaptığımız bir daha yapmıyoruz.*/
    OdaListeElemanlari.Add(newPlayer.UserId,entry);
}
```

Sonrasında ise oyuna başla butonunu aktif ediyoruz. Oda sahibinin başlat butonunun kontrolünü sağlamak için yaptığımız bir işlem.

```
OyunaBaslaButon.gameObject.SetActive(CheckPlayersReady());
```

5. Odadan oyuncu çıktığında

```
2 references
public override void OnPlayerLeftRoom(Player otherPlayer)
{
    /*Burası ise oyuncu odadan çıktığında çalışır. Eğer bu kısmı yapmazsak oyuncu satır objesi kalır aynı zamanda oyuncu listesi içerisinde de oyuncu kalır ve aslında böyle bir oyuncu olmadığı için oyun hiçbir zaman başlamaz*/
    Destroy(OyuncuListeElemanlari[otherPlayer.ActorNumber].gameObject);
    OyuncuListeElemanlari.Remove(otherPlayer.ActorNumber);
    /*Oyuncu çıktığında da oyuna başla butonunu kapatıyoruz.*/
    OyunaBaslaButon.gameObject.SetActive(CheckPlayersReady());
}
```

7)Sistem fonksiyonlarının yazılması | Tüm testlerin yapılması

Bir önceki bölümden devam ediyoruz. Sistem fonksiyonlarının yazılmasına devam ediyoruz ve son olarak genel test yapıp diğer işlemlerimize geçiyoruz. (!Sistem fonksiyonlarının tüm hali bu bölümün en altında bulunuyor)

1-Eğer oda kurucu değişirse

```
3 references
public override void OnMasterClientSwitched(Player newMasterClient)
{
    /*Eğer ki odayı kuran kişi değişirse bu kod tetiklenecektir.*/
    /*Neden bu fonksiyonu tetikleyeceğiz, odayı kuran kişi oyunu başlat butonunu da kullanması gerekiyor. Sonradan oyun kurucu olan kişinin başlat butonunu görebilmesi için kontrol yapacağız.*/
    if(PhotonNetwork.LocalPlayer.ActorNumber == newMasterClient.ActorNumber){
        /*Artık bu oyuncu oda kurucu demektir.*/
        OyunaBaslaButon.gameObject.SetActive(CheckPlayersReady());
    }
}
```

2-Eğer özelliklerde herhangi bir değişiklik olursa

```
4 references
public override void OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedProps)
{
    /*Her oyuncu birbirlerinin özelliklerini görebiliyordu, hazırım hazır değilim gibi. İşte burada özellikte herhangi bir değişiklik olursa bu buton tetiklenecek.*/

    /*Aşağıdaki bloğu yazma nedenimiz? tamamen bir otokontrol yani null gelme ihtimaline karşı yaptığımız bir kod*/
    if(OyuncuListeElemanlari == null){
        OyuncuListeElemanlari = new Dictionary<int, GameObject>();
    }

    /*Eğer bizim istediğimiz eleman tetiklendiyse biz aşağıdaki setPlayerReady çalıştırıyoruz.*/
    if(OyuncuListeElemanlari.TryGetValue(targetPlayer.ActorNumber,out GameObject entry)){
        if(changedProps.TryGetValue("IsPlayerReady", out object IsPlayerReady)){
            entry.GetComponent<PlayerListEntry>().SetPlayerReady((bool) IsPlayerReady);
        }
    }

    OyunaBaslaButon.gameObject.SetActive(CheckPlayersReady());
}
```


3-Kalan button işlemleri

1.Lobby den ana sayfaya dönen fonksiyon.

```
0 references
public void OnBackButtonClicked()
{
    /*Eğer lobby içerisindeyse lobby den çıkıp ana sayfaya dönmek istediğimize dair fonksiyon*/
    if(PhotonNetwork.InLobby)
    {
        PhotonNetwork.LeaveLobby();
        SetActivePanel(SecimPanel.name);
    }
}
```

2.Random bir odaya girmek istiyorsak.

```
0 references
public void OnJoinRandomRoomButtonClicked()
{
    //Random odaya girmek istiyorsak
    SetActivePanel(RandomodayagirPanel.name); //Random odaya giriliyor paneli
    //Random odaya girmemizi sağlayan kod
    PhotonNetwork.JoinRandomRoom();
}
```

3.Odayı terk etme butonu.

```
0 references
public void OnLeaveGameButtonClicked()
{
    PhotonNetwork.LeaveRoom();
}
```

4.Oyunu başlatma butonu

```
0 references
public void OnStartGameButtonClicked()
{
    //Artık oyun sahnesimize gideceğiz. Mevcut odanın görünebilirliğini ve açık kapalı olma durumunu kapatacağız.
    //Mevcut odanın çeşitli özelliklerine erişebilmemizi sağlıyor.
    PhotonNetwork.CurrentRoom.IsOpen = false;
    PhotonNetwork.CurrentRoom.IsVisible = false;
    PhotonNetwork.LoadLevel("Game"); //Game sahnesine geçiyoruz.
}
```

4-Oda ve Lobby işlemlerimizi yaptığımız GameManager.cs dosyamız.



GameManager.cs

7. BÖLÜM SONU “ODA VE LOBBY İŞLEMLERİNİN İSKELETİNİ TAMAMLADIK”

“SONRAKİ DERSLERDE OYUN İÇİ MANAGER İŞLEMLERİNİ DE GERÇEKLEŞTİRECEĞİZ.”

8)CountDownTimer yapımı | Oyun başlatma işlemlerinin yapımı

Artık burada oyuna katıl dediğimizde oyuncuların katılablmesini ve oda kurucunun oyunu başla dediğinde oyunun başlatılablmesini gerçekleştireceğiz. Ve son olarak CountDownTimer geri sayım sistemini göreceğiz.

CountDownTimer nedir?

CountDownTimer araba yarışı oyunlarını hayal et oyun başlamadan önce 3 ten geriye doğru saymaya başlar. Oyuncuların hazırlanması amaçlanır. Biz de bu işlemin yapımını göreceğiz.

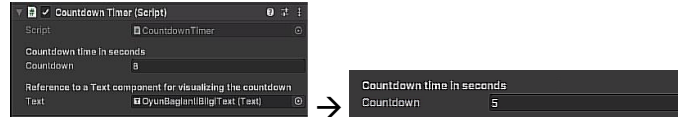
Burada iki seçeneğimiz mevcut 4 oyuncu olduğunu düşünelim 4. Oyuncu katılır katılmaz oyunu başlatabiliriz. Ya da CountDownTimer kullanarak 4 oyununcun geldiğinde geri sayım başlayıp geri sayım biter bitmez oyunun başlatılması. Biz burada ikinci seçeneğin yapımını gerçekleştireceğiz.

Bu işlemin teknik tarafı ise, oyunumuz online olduğu için her oyuncunun internet bağlantısı aynı değildir. Yükleme işleminin bağlantılardan dolayı gecikmelerin de önüne geçebilmek için tüm oyuncular oyuna bağlandığında başlatacağız.

Ayrıca bu sayacı serverdan alacağımız için tüm oyuncular senkron bir şekilde saniyeyi görecekler.

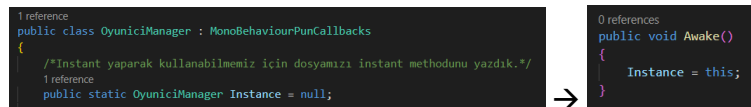
1-İlk olarak Game sahnemizi açıyoruz. (Oyun sahnesi)

1.GameManager isminde bir boş objen yok ise oluşturup, Countdown Timer script ekliyoruz. Bu scriptin photon'dan alıyoruz. Bu dosyamızı ana script dosyasından tetikleyeceğiz. Burada tek yapmamız gereken geri sayım saniyesini vermemiz bizim için yeterli olacaktır.

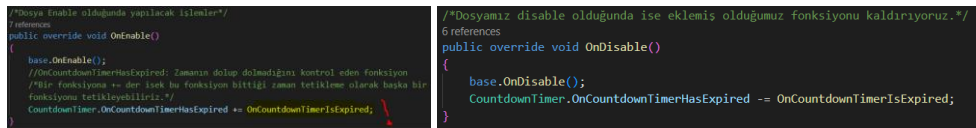


2. Oyuncumuza ait hazırlamış olduğumuz script dosyasını açıyoruz. Önceki bölümlerde görmüştük.

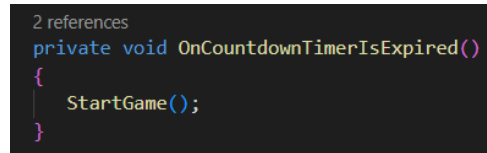
1..İlk olarak script dosyamıza erişebilmek için instance methodunu yazık



2..Photon bize sunmuş olduğu Enable() komutunu yazacağız. Bu komut dosyamız ortaya çıktığında bu işlemleri yap demek anlamına gelir.



Geri sayım bittiğinde çalışacak fonksiyonun yazımı. !Oyunu başlatıyoruz. Zaman bittiğinde StartGame fonksiyonunu tetikletiyoruz.



3..Sonrasında ise start methodunda zaman bittiğinde tüm oyuncuların orada olup olmadığının kontrolünün yapılacağı için burada bir oyunculara özellik tanımladık.

```
0 references
public void Start()
{
    /*Oyun başlar başlamaz sahneye gelip gelmediklerini
    anlayabilmek için bir özellik tanımlayacağız.*/
    Hashtable props = new Hashtable{
        {"PlayerLoadedLevel",true}
    };
    PhotonNetwork.LocalPlayer.SetCustomProperties(props);
}
```

4..Şimdi ise bir üstte tanımladığımız özelliği kontrol etme işlemini gerçekleştirelim.

```
0 references
private bool CheckAllPlayerLoadedLevel()
{
    // Oyuncular sahneye geldi mi gelmedi mi burada kontrol edilecek döngü
    foreach (Player p in PhotonNetwork.PlayerList)
    {
        if(p.CustomProperties.TryGetValue("PlayerLoadedLevel", out object playerLoadedLevel)){
            //Objeye olarak aldığımız veriyi bool çeviriyoruz.
            if((bool) playerLoadedLevel){
                continue;
            }
            return false;
        }
    }
    return true;
}
```

5..Zamanı başlatma işlemi

```
0 references
public override void OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedProps)
{
    // Oyuncu özelliklerinde bir değişim olursa burada yakalanacak
    //Odaya giren oyuncunun yeni özellik eklenmesi için yine burası çalışacak, burada oyuncuların
    gelip gelmediğini anlayacağız fonksiyonu çağıracağız.*/
    //Oda kurucu oyuncu sonucu giren oyuncu olmaz ondan dolayı eğer bu oyuncu oda kurucu değilse devam et
    if(!PhotonNetwork.IsMasterClient){
        return;
    }

    bool StartTimeSet = CountdownTimer.TryGetStartTime(out int startTime);
    //Eğer özellik değişen oyuncunun içerisinde Check... fonksiyonumuz true dıerse tüm oyuncular gelmiş demektir.
    if(changedProps.ContainsKey("PlayerLoadedLevel")){
        if(CheckAllPlayerLoadedLevel()){
            //Eğer tüm oyuncular hazır ise, zamanı başlatıyoruz.
            if(StartTimeSet){
                CountdownTimer.SetStartTime();
            }
        }
        else{
            InfoText.text = "Diğer oyuncular bekleniyor...";
        }
    }
}
```

6..StartGame

```
0 references
private void StartGame()
{
    Debug.Log("StartGame!");
    //Zaman bitince OnCountdownTimerIsExpired fonksiyonu çağırılıyor. Artık oyuncu oluşturuyoruz ve oyun başlıyor.*/
    PhotonNetwork.Instantiate("OyuncuOluşturmaObjesi", Vector3.zero, Quaternion.identity, 0);
}
```

7..Diğer fonksiyonlar

```
#region PUN_CALLBACKS

8 references
public override void OnDisconnected(DisconnectCause cause)
{
    SceneManager.LoadScene("Lobby");
}

5 references
public override void OnLeftRoom()
{
    PhotonNetwork.Disconnect();
}

2 references
public override void OnPlayerLeftRoom(Player otherPlayer)
{
    Debug.Log("Odadan Oyuncu Çıktı");
}
```

8..OyuniçiManager dosyamız



OyuncuManager.cs

9)Oda katıl butonunun kontrol işlemlerinin yapılması

Bu bölümdeki yapacaklarımız ise, kapasitesi dolu olan odaların butonun aktifliğini kapatmamız gerekiyor.



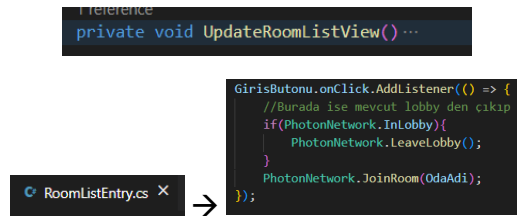
Soru: Oda neden hala açık kalıyor? Diğer bölümlerde odayı kapatmıştık?

Biz odayı kapalı olma durumunu kurucu oyuncu oyunu başlat dediğinde kapatıyorduk. Oyunu başlatmadan önce herhangi bir kapatma yapmıyoruz.

Yukarıdaki nedenden dolayı kapasite kontrolünü yapıp kapasitesi dolan odalarda kapat butonunun aktifliğini kapatacağız. Eğer bir oyuncu o odada oynamaktan vazgeçerse eğer bu sefer bu aktifliği geri açacağız. Oyuncuların tekrar girebilmesini sağlayabilmek için.

1-İlk olarak, GameManager.cs dosyamıza gidiyoruz. Bizim oda bilgimizi, yani listemizi veren fonksiyonumuz UpdateRoomListView perki buradaki butonumuzu nerde listener veriyoruz. O ise RoomListEntry.cs klasörümüzde.

Burada bir kontrol yapıp eğer oda dolu ise kapatıp, oda eğer müsait olduğu taktirde ise açacağız.



2- Bu işlemimizi ise, UpdateRoomListView fonksiyonumuzdan bir state değişkeni göndereceğiz. Bu değişkeni RoomListEntry de kontrolümüzü sağlayacağız.

GameManager.cs

```
void UpdateRoomListView()
{
    //Olistendığımız gameci listeleri kullanarak odaları satır satır yazmamızı sağlar.
    foreach (RoomInfo info in OdaCacheList.Values)
    {
        //Burada ne kadar satır var ise o kadar liste oluşturmamız ve
        //aldığımız bilgileri ile o objenin içeriğini dolduruyoruz. */
        //Photon Instantiate komutunu kullanmamızın sebebi bizim
        //satır listeyi photon gameci olarak vermesinden kaynaklı yani
        //tekrardan birim zamanı ile o objenin içeriğini dolduruyoruz. */
        GameObject entry = Instantiate(OdaListesiPrefab);
        //Olistendığımız entry objesinin içeriğini oyuncağı olarak obje olarak atıyoruz
        entry.transform.SetParent(OdaListesiContent.transform);
        //Scale değeriyle herhangi bir kayıp olmasını istemiyoruz.
        entry.transform.localScale = Vector3.one;
        //Olistendığımız oda listesi elemanı da kendisine ait hazırlandığına
        //listenin içeriğine atıyoruz */
        //info.MaxPlayers == info.PlayerCount) odayı aktif ise true, değil ise false döndürüyoruz.
        entry.GetComponent<RoomListEntry>().Initialize(info.Name, (byte) info.PlayerCount, info.MaxPlayers, (info.MaxPlayers == info.PlayerCount));
        OdaListesiEmanlari.Add(info.Name, entry);
    }
}
```

RoomListEntry.cs

```
public void Start()
{
    //Giris butonu bir örnek obje olduğu için bir onun click methodu boş geliyor.
    //Start methodunda event vererek click yapma işlemini tekrardan kazandırmış oluyoruz.
    if (!RoomState)
    {
        GirisButonu.interactable = true;
        GirisButonu.onClick.AddListener(() =>
        {
            //Burada ise mevcut lobby den çıkıp odaya bağlama işlemini yaptığını belirtiyoruz.
            if (PhotonNetwork.InLobby)
            {
                PhotonNetwork.LeaveLobby();
            }
            PhotonNetwork.JoinRoom(OdaAdi);
        });
    }
    else
    {
        GirisButonu.interactable = false;
    }
}
```

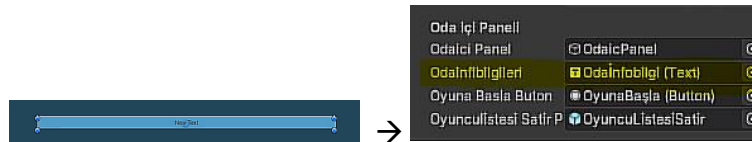
10) İstatistik bölümlerinin yapılması

Bu bölümümüzde ise, örneğimizi detaylandıracağız.

1-Odayı kurduktan sonraki ekranımızda odanın iç bilgilerini de vereceğiz. Yani odayı kuran, kaç kişi olduğu vs. istatistik alanı tutacağız.



1. İstatistik bölümü için bir panel oluşturuyoruz. Ve oda bilgilerini içeren bir text oluşturuyoruz.



2. GameManager.cs dosyamıza gidiyoruz.

```
public override void OnJoinedRoom()
{
    /*Artık odaya girdiğimiz için buradaki listeye ihtiyacımız kalmadı.
    İhtiyacımız olmadığı için sildiğimizde ön bellekte kapladığı yeri
    temizlemiş oluruz. Perf. kazancı sağlıyoruz.*/
    OdaCachelist.Clear();
    SetActivePanel(OdaiciPanel.name);
}

entry.GetComponent<PlayerListEntry>().Initialize(p.ActorNumber,p.NickName);

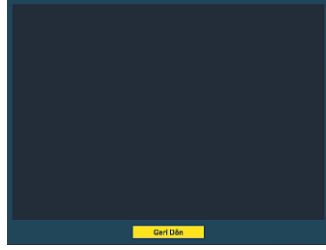
Odainfobilgileri.text = "Odanın adı : " + PhotonNetwork.CurrentRoom.Name + " Odayı kuran kişi : "
    + PhotonNetwork.MasterClient.NickName + " Mevcut Oyuncu : " + PhotonNetwork.CurrentRoom.PlayerCount
    + " Maksimum Oyuncu : " + PhotonNetwork.CurrentRoom.MaxPlayers;
```

3. Bu yazmış olduğumuz satırı, odaya giriş yaptığımızda gösterdik. Yeni bir oyuncu girdiğinde ya da bir oyuncu çıktığında bu satırları güncellememiz gerekiyor. Kısacası oyuncunun girip, çıktığı her yerde bu text güncellememiz gerekiyor.

```
2 references
public override void OnPlayerEnteredRoom(Player newPlayer)
{
    /*Eğer ki odaya yeni bir oyuncu girerse, oda liste elemanlarına ve satırına bu kullanıcıyı eklemem gerekiyor.*/
    /*Eğer bu işlemi yapmazsa oyuncular bağlantı yaptığında birbirlerini görmeyebilir. Ya da görmez!*/
    GameObject entry = Instantiate(OyuncuListesiSatirPrefab);
    entry.transform.SetParent(OdaiciPanel.transform);
    entry.transform.localScale = Vector3.one;
    entry.GetComponent<PlayerListEntry>().Initialize(newPlayer.ActorNumber,newPlayer.NickName);
    //Burada hazırım butonu entegrasyonu yapmamaızın sebebi zaten oluşturma işlemini yaptığımız bir daha yapmıyoruz.
    OdaListeElemanlari.Add(newPlayer.UserId,entry);
    OyunaBaslaButon.gameObject.SetActive(CheckPlayersReady());
    Odainfobilgileri.text = "Odanın adı : " + PhotonNetwork.CurrentRoom.Name + " Odayı kuran kişi : "
        + PhotonNetwork.MasterClient.NickName + " Mevcut Oyuncu : " + PhotonNetwork.CurrentRoom.PlayerCount
        + " Maksimum Oyuncu : " + PhotonNetwork.CurrentRoom.MaxPlayers;
}

2 references
public override void OnPlayerLeftRoom(Player otherPlayer)
{
    /*Burası ise oyuncu odadan çıktığında çalışır. Eğer bu kısmı yapmazsak oyuncu satır objesi kalır aynı
    zamanda oyuncu listesi içerisinde de oyuncu kalır ve aslında böyle bir oyuncu olmadığı için
    oyun hiçbir zaman başlamaz*/
    Destroy(OyuncuListeElemanlari[otherPlayer.ActorNumber].gameObject);
    OyuncuListeElemanlari.Remove(otherPlayer.ActorNumber);
    //Oyuncu çıktığında da oyuna başla butonunu kapatıyoruz.
    OyunaBaslaButon.gameObject.SetActive(CheckPlayersReady());
    Odainfobilgileri.text = "Odanın adı : " + PhotonNetwork.CurrentRoom.Name + " Odayı kuran kişi : "
        + PhotonNetwork.MasterClient.NickName + " Mevcut Oyuncu : " + PhotonNetwork.CurrentRoom.PlayerCount
        + " Maksimum Oyuncu : " + PhotonNetwork.CurrentRoom.MaxPlayers;
}
```

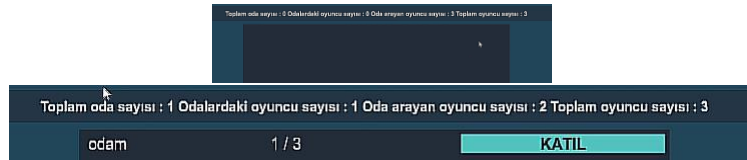
2-Bir de oda listesi dediğimizde tüm odaları görebiliyorduk. Buradaki bilgileri vereceğiz. Kaç tane oda var, kaç oyuncu oyun oynuyor, kaç oyuncu oda bekliyor, toplam kaç oyuncu var gibi genel istatistik alanı yapacağız.



1.Burada da bir üstteki bilgi kısmı gibi bir panel ve text hazırlıyoruz. Ve dışarıdan o text alıyoruz.

2.GameManager.cs dosyasına giriyoruz

```
public void OnRoomListButtonClicked()
{
    if(!PhotonNetwork.InLobby){
        PhotonNetwork.JoinLobby();
    }
    SetActivePanel(OdalistePanel.name);
    //Sürekli bir invoke methodu çalıştıracak isek InvokeRepeating olarak belirtmemiz gerekiyor.
    //method adı, ne zaman başlaması gerektiği, kaç saniyede bir tekrar etmesi gerektiğini belirtiyoruz.
    InvokeRepeating("istatistikGetir",0,5);
}
/*İstatistikleri sürekli alabilmek için bir invoke methodu kullanacağız.
Coroutineler de kullanabiliriz ancak bu durum bize sorun yaratabilir. Photon ile çakışmalar
yaşanabilir*/
0 references
void istatistikGetir(){
    //Odaliste paneli aktif olduğu durumlardda
    if(OdalistePanel.activeSelf){
        GenelInfoBilgileri.text = "Toplam oda sayısı : " + PhotonNetwork.CountOfRooms +
        "Odalardaki oyuncu sayısı : " + PhotonNetwork.CountOfPlayersInRooms +
        "Oda arayan oyuncu sayısı : " + PhotonNetwork.CountOfPlayersOnMaster +
        "Toplam oyuncu sayısı : " + PhotonNetwork.CountOfPlayers;
    }
    else{
        //Eğer panel kapalı ise invoke methodunu kapatıyoruz
        CancelInvoke("istatistikGetir");
    }
}
```



11. BÖLÜM SONU “ODA VE LOBBY İŞLEMLERİNİ TAMAMLADIK”

“SONRAKİ DERSLERDE EXTRA OLARAK ODALARI ŞİFRELEME İŞLEMLERİ GERÇEKLEŞTİRECEĞİZ.”

11)Şifreli oda yapma | Özel oda yapma

Bu dersimizde oda kurucu isterse şifre koyabilir.

1-Şifreli oda yapmak için ilk olarak bir şifre belirlememiz adına oda kurma esnasında şifre inputu koyuyoruz.



2-GameManger.cs dosyamızdan dışarıdan şifre text alacağımızı söylüyoruz.

```
public InputField OdaSifresiInput;
```

3- Sonrasında ise oda kurma fonksiyonumuza gidiyoruz.

```
0 references
public void OnCreateRoomButtonClicked()
{
    string roomName = OdaAdiInput.text;
    roomName = (roomName.Equals(string.Empty)) ? "Oda " + Random.Range(100, 100000) : roomName;
    //MaxPlayer değeri türü byte olduğunu unutma
    byte.TryParse(MaksimumOyuncuInput.text, out byte MaxPlayer);
    MaxPlayer = (byte) Mathf.Clamp(MaxPlayer, 2, 8);

    /*Oda kurumu esnasında yeni bir şifre özelliği oluşturduk. Eğer şifre var ise şifre değeri yok
    ise 0 değerini alacak ve odaya girişlerde bunun kontrolü sağlanarak içeri giriş yapılabilir.*
    string[] LobbyOptions = new string[1];
    LobbyOptions[0] = "sifre";

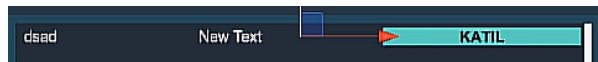
    int sifre = OdaSifresiInput.Equals(string.Empty) ? 0 : int.Parse(OdaAdiInput.text);

    Hashtable props = new Hashtable{
        {"sifre", sifre}
    };

    RoomOptions options = new RoomOptions{
        MaxPlayers = MaxPlayer,
        CustomRoomPropertiesForLobby = LobbyOptions,
        CustomRoomProperties = props
    };

    //Optionsları kullanabilmek için TypedLobby.Default olarak girmemiz gerekiyor.
    PhotonNetwork.CreateRoom(roomName, options, TypedLobby.Default);
}
```

4-Sıradaki işlemimiz ise bu özellikleri artık kullanacağız. İlk önce ilgili odanın şifresini kontrol edeceğiz. Eğer oda şifreli ise ilk olarak oda listelerinde şifreli olduğunu belirteceğiz.



Sonrasında RoomListEntry.cs dosyasına gidiyoruz. Ve dışarıdan bir text inputu alacağını söylüyoruz. Ve ikinci olarak GameManager.cs dosyasından instance methodu aracılığı ile gelecek şifreyi alabilecek bir int değişken oluşturuyoruz.

```
Assets > Photon > PhotonUnityNetworking > Demos > DemoAsteroids > Son...
1 reference
9 public class RoomListEntry : MonoBehaviour
10 {
11     1 reference
    public Text OdaAdiText;
12     1 reference
    public Text OdadakiOyuncuSayisiText;
13     3 references
    public Button GiriButonu;
14     0 references
    public Text OdaSifresiText;
15
16     2 references
    private string OdaAdi;
17     2 references
    private bool RoomState;
18     1 reference
    private int Password;
    0 references
}
```

5-Sıradaki işlemimiz ise GameManager.cs dosyamıza gidiyoruz. UpdateRoomListView methodunda listleri oluşturma için yazdığımız fonksiyon mevcut. Bunun içerisine gidiyoruz sonrasında ilk önce şifre var mı yok mu kontrolü yapıp, sonrasında şifreyi RoomListEntry.cs dosyasına gönderiyoruz.

EXTRA YÖNTEM

```
/*! Eğer şifre 0 atamadığımızı düşünelim. Şifre verildiği takdirde bu özellik ekleniyor ise, aşağıdaki gibi bu özelliği atarak eğer
var ise şifreli yok ise şifresiz olduğunu belirtebilirsin. EXTRA YÖNTEM*/
if(info.CustomProperties.ContainsKey("şifre")){
}
else{
}
Odalistelemanlari.Add(info.Name,entry);
```

Bizim kullandığımız yöntem

```
private void UpdateRoomListView()
{
    //Oluşturduğumuz güncel listeleri kullanarak odaları satır satır yazmamızı sağlar.
    foreach (RoomInfo info in OdaCacheList.Values)
    {
        /*Burada ne kadar satır var ise o kadar liste oluşturuyoruz ve
        aldığımız bilgiler ile o objenin içerisini dolduruyoruz.*/

        /*Photon Instantiate komutunu kullanmamamızın sebebi bizim
        zaten listeyi photon güncel olarak vermesinden kaynaklı yani
        tekrardan bizim yanuca tarafı liste oluşturmamıza gerek yok
        */
        GameObject entry = Instantiate(OdalistesisiSatiPrefab);
        //Oluşturduğumuz entry odalistesiconent içerisinde çocuk obje olarak atıyorum
        entry.transform.SetParent(Odalistesiconent.transform);
        //Scale değeriinde herhangi bir kayıp olmasını istemiyoruz.
        entry.transform.localScale = Vector3.one;
        /*Oluşturduğumuz oda listesi elemanı da kendisine ait hazırladığımız
        listemin içerisine atıyoruz.*/
        /*info.MaxPlayers = info.PlayerCount) eğer exit ise true, değil ise false döndürecek*/
        info.CustomProperties.TryGetValue("şifre", out object şifre);

        entry.GetComponent<RoomListEntry>().Initialize(info.Name, (byte) info.PlayerCount, info.MaxPlayers, (info.MaxPlayers == info.PlayerCount), (int) şifre);
        Odalistelemanlari.Add(info.Name,entry);
    }
}
```

6-Artık şifreyi RoomListEntry.cs dosyasına gönderebiliyoruz. Sıradaki işlemimiz ise şu, instance aldığımız şifreyi oluşturduğumuz değişkene atıyoruz.

```
1 reference
public void Initialize(string odaadi, byte MevcutOyuncu, byte MaksimumOyuncu, bool state, int şifre)
{
    OdaAdi = odaadi;
    OdaAdiText.text = odaadi;
    OdadakiOyuncuSayisiText.text = MevcutOyuncu + " / " + MaksimumOyuncu;
    RoomState = state;
    Password = şifre;
}
```

7-Sonrasında, şifre olup olmadığının kontrolünü yapıyoruz. Eğer şifre var ise text kısmına şifreli yok ise şifresiz olduğunu belirtiyoruz.

```
1 reference
public void Initialize(string odaadi, byte MevcutOyuncu, byte MaksimumOyuncu, bool state, int şifre)
{
    OdaAdi = odaadi;
    OdaAdiText.text = odaadi;
    OdadakiOyuncuSayisiText.text = MevcutOyuncu + " / " + MaksimumOyuncu;
    RoomState = state;
    Password = şifre;
    OdaSifresiText.text = şifre != 0 ? "Şifreli" : "Şifresiz";
}
```


12)Şifreli oda yapma | Özel oda yapma

Artık son bölüme geçmiş bulunmaktayız. Burada odaya katıl butonunda bir şart belirleyeceğiz. Eğer şifre yok ise direkt normal olarak odaya geçiş sağlanabilecek. Ancak eğer şifre var ise ne yapacağız bir canvas çıkarıp şifresi girilip, şifreyi doğru girdiği taktirde odadan içeriye alacağız.

1-İki adet panel oluşturuyoruz. Biri şifre paneli biri şifre girildiğinde yanlış girilir ise, hata mesajını kullanıcıya bildirecek panel.



2-Bu panelleri dışarıdan almıyoruz. Odanın satırı prefab türetildiği için dışarıdan kod ile alacağız.

```
public void Start()
{
    /*
    Giriş butonu bir örnek obje olduğu için biz onun click methodu boş geliyor.
    Start methodunda event vererek click yapma işlemini tekrardan kazandırmış oluyoruz.
    */
    if (!RoomState)
    {
        GirisButonu.interactable = true;
        GirisButonu.onClick.AddListener(() =>
        {
            if (Password != 0)
            {
                şifrepanel = GameObject.Find("OdaListesiPanel").transform.Find("SifrePaneli").gameObject;
                şifrepanel.SetActive(true);
                şifrepanel.transform.Find("GirisButon").GetComponent<Button>().onClick.AddListener(() => SifreKontrol(OdaAdi));
                şifrepanel.transform.Find("KapatButon").GetComponent<Button>().onClick.AddListener(() => pencerekapat(şifrepanel));
            }
            else
            {
                //Burada ise mevcut lobby den çıkıp odaya bağlanma işlemini yaptığını belirtiyoruz.
                if (PhotonNetwork.InLobby)
                {
                    PhotonNetwork.LeaveLobby();
                }
                PhotonNetwork.JoinRoom(OdaAdi);
            }
        });
    }
    else
    {
        GirisButonu.interactable = false;
    }
}

1 reference
public void SifreKontrol(string odaadi){
    if(şifrepanel.GetComponentInChildren<InputField>().text == Password.ToString()){
        //Sifre doğru giriş yapıldı
        if (PhotonNetwork.InLobby)
        {
            PhotonNetwork.LeaveLobby();
        }
        PhotonNetwork.JoinRoom(OdaAdi);
    }
    else{
        //Sifre hatalı
        hatapanel = GameObject.Find("OdaListesiPanel").transform.Find("SifreYanlis").gameObject;
        hatapanel.SetActive(true);
        hatapanel.GetComponentInChildren<Button>().onClick.AddListener(() => pencerekapat(hatapanel));
    }
}

2 references
void pencerekapat(GameObject gameObject){
    gameObject.SetActive(false);
}
```

13)Genel bu bölümde kullanılan tüm dosyalar

