PHOTON ÇEŞİTLİ YÖNTEM VE İŞLEMLER

1)Odalara özellik ekleme | Room Properties | Room CustomRoomPropertiesForLobby

Odalarımız oyuncuları barındırdığı için daha geniş özelliklere ihtiyaç duyabiliyor. Örneğin bazı oyunlarda çeşitli odalar bulunmaktadır bunlar altın oda, gümüş oda ya da vip oda gibi… Odalara özgü yeni özellikler ekleme ve bunların kontrolünü bu kısımda göreceğiz.

1-Bu dersimizi aşağıdaki .cs dosyasından takip edeceğiz.

GameManager1.cs

2-Eğer biz özelliklerimizi lobby'den de erişmek istiyorsak, o zaman bir string tanımlayıp sırası önemli o keyleri string içerisinde de barındırmamız gerekiyor.

```
public void OdaOlustur()
{
    string roomName = OdaAdiInput.text;

    string[] LobbyOptions = new string[1];
    LobbyOptions[0] = "key";

    Hashtable props = new Hashtable(){
        {"key","value"},
    };
    PhotonNetwork.CreateRoom(roomName,new RoomOptions{MaxPlayers = 3},TypedLobby.Default);
}
```

3-Oda özelliklerini girmek

```
0 references
public void OdaOlustur()
{
    string roomName = OdaAdiInput.text;
    //Özellikleri burada belirtiyoruz.
    Hashtable props = new Hashtable(){
        {"key","value"},
    };
    //Lobby den çekmek istediğimiz özelliklerin keylerini veriyoruz.
    string[] LobbyOptions = new string[1];
    LobbyOptions[0] = "key";

    //Yeni bir options üretiyoruz hem özellikleri hem de lobby özellik keylerini veriyoruz.
    RoomOptions options = new RoomOptions{
        MaxPlayers = 3,
        //CustomRoomPropertiesForLobby --> Lobyden herhangi bir odanın özelliğine erişmek istiyorsam o anahatarı vereceğiz.
        CustomRoomPropertiesForLobby = LobbyOptions,
        //CustomRoomProperties --> Oda özelliğini ise buradan veriyoruz.
        CustomRoomProperties = props
    };
    PhotonNetwork.CreateRoom(roomName,options,TypedLobby.Default);
}
```

```
/*
Kısa bir özet geçmek istiyorum.
Eğer biz odanın içerisindeysek, oluşturduğumuz props devreye girecek
Eğer biz lobby de isek, oluşturduğumuz lobby options devreye girecek
*/
```

4-Eklenen özellikleri vs. nasıl okuyabiliriz?

1.Oda içerisinde veri okuma

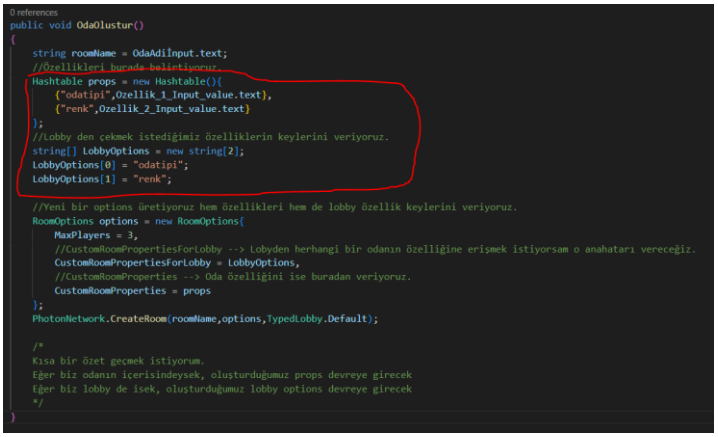
```
public override void OnJoinedRoom()
{
    SetActivePanel(OdaicPanel.name);
    PhotonNetwork.CurrentRoom.CustomProperties.TryGetValue("key", out object prop);
    OdainfobilgileriText.text = "Odanın Adı : " + PhotonNetwork.CurrentRoom.Name + " | Odayı kuran kişi :  " + PhotonNetwork.MasterClient.NickName
    + " | Key(Properties) : " + (string) prop;
}
```

2.Lobby içerisinde veri okuma

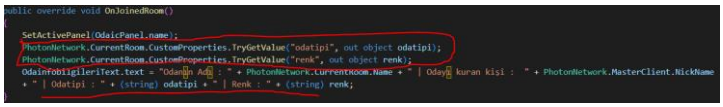
```
private void UpdateCachedRoomList(List<RoomInfo> roomList)
{
    foreach (RoomInfo info in roomList)
    {
        info.CustomProperties.TryGetValue("key", out object prop);
        OdalistesiText.text = "Oda adı: " + info.Name + "Key(Properties)" + (string) prop;
    }

}
```

2)Odalara özellik ekleme 2 | Room SetCustomProperties

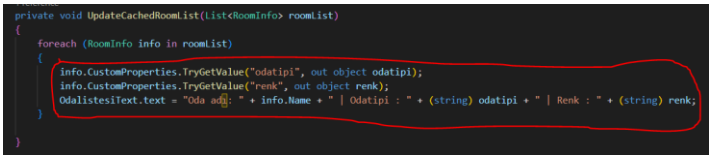
1-Şimdi imputlardan gelen değerleri alıp value değerlerine set edeceğiz ve bu kısımı bitireceğiz.



1.Odadayken veri çekme

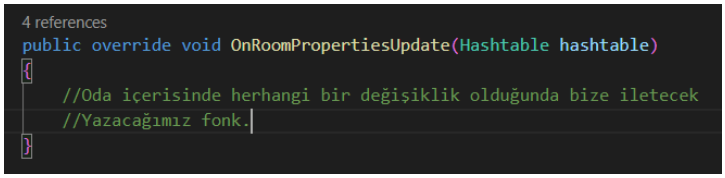


2.Lobbydeyken veri çekme



2-Son olarak mevcut oda özelliğini değiştirme işlemini gerçekleştirelim.

3-Oda listelerinde herhangi bir değişikte tetiklenen fonksiyonumuz mevcuttu. Aynı şekilde oda içerisinde herhangi bir değişen olursa o an tetiklenen fonksiyon da mevcut. O fonksiyonu da kullanabiliriz.



4-GameManager son hali

GameManager1.cs

3)Takım yönetimi | Oyuncuları takıma dahil etme ve diğer takım işlemleri

1- Online oyunlarda ihtiyaç duyulabilen takım kurma sistemini inceleyeceğiz. Photon bu imkanı bize sunuyor. Eğer oyununuzda takım kurma amaçlanıyor ise bu yapıyı kullanabiliriz.

Eski sahnemize gidiyoruz.

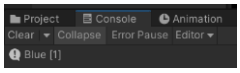
1.Oyuncu scripti içerisinde Start methodunda işlemlerimize başlıyoruz.

1..Oyuncuyu takıma dahil etme. Hangi oyuncuyu hangi takıma dahil edeceğimize ait yapıyı da göreceğiz. Yani oyuncuları takımlara bölme işlemini göreceğiz. !LocalPlayer: Oyuncunun kendisini almamızı sağlar.

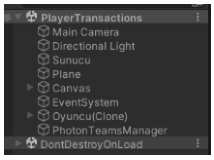
```
/*JoinTeam iki takım olarak nitelendirilir. Mavi takımın kodu 1, Kırmızı takımın kodu ise 2*/
PhotonNetwork.LocalPlayer.JoinTeam(1);
```

2..Oyuncunun hangi takımda olduğunu nasıl alıyoruz.

```
//is.mine komutu içerisine yazıyoruz. Çünkü diğer oyuncuları etkilemek istemiyoruz.
if(Input.GetKeyDown(KeyCode.T))
    Debug.Log(PhotonNetwork.LocalPlayer.GetPhotonTeam());
```



Eğer takım sistemini kullanıyorsak hierarchy bölümünde PhotonTeamsManager objesini görüyorsak sorun yok demektir.



3..Takım sistemini derinlemesine götürmek istersek yani, oyuncunun istersek takım kodunu ya da takım adını alabiliriz.

```
Debug.Log(PhotonNetwork.LocalPlayer.GetPhotonTeam().Name);
Debug.Log(PhotonNetwork.LocalPlayer.GetPhotonTeam().Code);
```

2-Oyuncuları takımlara nasıl dağıtacağız. Bu işlemleri sunucu yönetimi scriptinde gerçekleştireceğiz.

Eğer senin takım dağıtma ile ilgili bir fonksiyonun varsa ya da matematiğin var ise onu kullanabilirsin. Ya da birinci giren örneğin mavi ikinci giren kırmızı üçüncü mavi diye ayırarak gidebilirsin. Ya da oyuncularının seviyesi var ise seviyelerine göre takımlara ayırabilirsin gibi…

Bir örnek ile pekiştirelim, takımlara dahil etme işini. Kendi methodun – algoritman ile de bu işlemi genişletebileceğini unutma.

Örneğimiz o an odamızda kaç oyuncu var ve oyuncu sayısına göre bir mavi bir kırmızı takıma ayırma işlemini göreceğiz.

Sunucu Yönetim -/- oyuncu

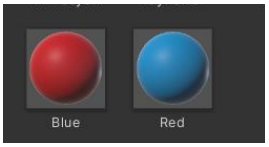
```
5 references
public override void OnJoinedRoom()
{
    panel.SetActive(false);

    /*
    PhotonNetwork.PlayerList.Length --> oyuncu listesindeki sayıyı alıyoruz (Tek bir odan varsa)
    PhotonNetwork.CurrentRoom.PlayerCount --> Mevcut oda içerisindeki oyuncu sayısını alıyoruz. (Birçok odan varsa) */
    int kalan = PhotonNetwork.PlayerList.Length % 2; //Oyuncu sayısının kalanını alıyor ve kalanını bize veriyor.
    //Kalan 1 ise mavi kalan 0 ise kırmızı takıma vereceğiz oyuncuyu
    byte teamNo;
    if(kalan == 1)
        teamNo = 1;
    else
        teamNo = 2;

    GameObject oyuncu = PhotonNetwork.Instantiate("Oyuncu", Vector3.zero, Quaternion.identity);
    oyuncu.GetComponent<PhotonView>().RPC("ChooseTeam",RpcTarget.All,teamNo);
    Debug.Log("Odaya Girildi.");
}
```

```
[PunRPC]
0 references
void ChooseTeam(byte teamNo){
    PhotonNetwork.LocalPlayer.JoinTeam(teamNo);
}
```

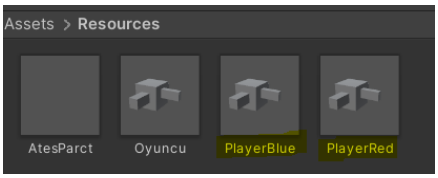
4)Takım yönetimi2

1-Şimdi hangi takımda olduklarını daha iyi görebilmek için takımlara ait renkler atayacağız.

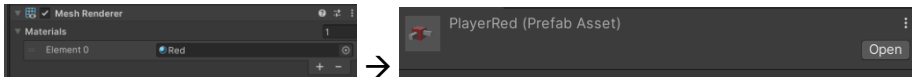
1.Renklerimizi hazırlıyoruz.



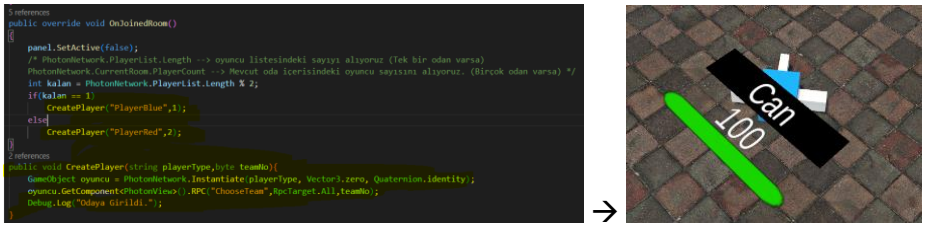
2.Resources klasöründeki oyuncumuzu türettik ve birine playerblue diğerine ise playerred isimlerini verdik.



3.Bu renkleri isimlerine göre atıyoruz.



4. Şimdi ise takımlarına göre oyuncuları oluşturma



2- Takım yönetimi için kullanabileceğimiz en temel kullanabilir yöntemler mevcut. Gelin onlara bakalım.

```
//is.mine komutu içerisine yazıyoruz. Çünkü diğer oyuncuları etkilemek istemiyoruz.
//Mevcut takımını alıyoruz.
if(Input.GetKeyDown(KeyCode.T))
    Debug.Log(PhotonNetwork.LocalPlayer.GetPhotonTeam());
//Mevcut takımından ayrılmak isterse
if(Input.GetKeyDown(KeyCode.Y))
    PhotonNetwork.LocalPlayer.LeaveCurrentTeam();
//Oyuncunun takım değiştirme işlemi yapımı
if(Input.GetKeyDown(KeyCode.U)){
    //Lambda if kullanımı eğer takım kodu 1'e eşit ise 2 değil ise 1 yapacak
    byte team = (byte) (PhotonNetwork.LocalPlayer.GetPhotonTeam().Code == 1 ? 2 : 1);
    PhotonNetwork.LocalPlayer.SwitchTeam(team);
    //gameObject.GetComponent<MeshRenderer>().material.color = team == 1 ? Color.blue : Color.red;
    if(team == 1){
        gameObject.GetComponent<MeshRenderer>().material.color = Color.blue;
        gameObject.name = "PlayerBlue";
    }
    else{
        gameObject.GetComponent<MeshRenderer>().material.color = Color.red;
        gameObject.name = "PlayerRed";
    }

}
//Takım arkadaşlarımın listesini almak istersek
if(Input.GetKeyDown(KeyCode.I)){
    PhotonNetwork.LocalPlayer.TryGetTeamMates(out Player[] friends);

    foreach (var a in friends)
    {
        Debug.Log("Friends Name: " + a.NickName);
    }
}
```

5)WEBHOOKS & WEBRPC | Uzak sistemlere sorgu gönderme | Genel anlatım ve kurulum

WEBRPC: Harici hizmetleri yani kendimize ait sistemleri photon ile entegre etmenin en esnek yoludur. Örneğin uzak bir sistemden veri almak istiyorsan Webrpc ile bunu gerçekleştirebiliriz.

Oyuncuların verilerini bazı bilgilerini veri tabanında tutuyor olabiliriz ya da bize ait bir sistemden belirli sorgulamalar yaparak oyunumuza veri çekmek istiyor olabiliriz. Weprpc bize bu imkanı sağlıyor.

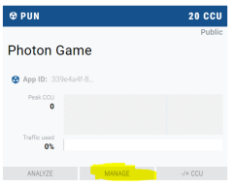
Webrpc kullanabilmek için;

- İlk olarak bir sistemimiz yani websitemiz olması gerekiyor.

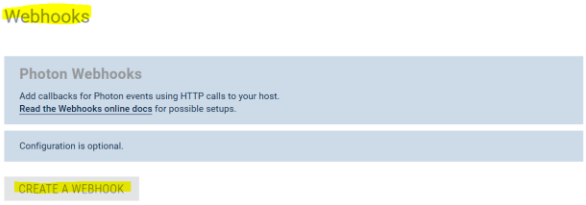
- İkinci olarak ise photon yönetim panelinden webhooks ayarlarını yani web kancaları ayarlarını yapmamız gerekiyor.

Webhooks ayarları:

1-PhotonNetwork içerisine gidiyoruz. Photon network içerisinde oluşturduğumuz uygulama sekmemizin içerisinde manage butonuna basıp oluşturmuş olduğumuz pan hizmetini yönetmek istediğimizi bildiriyoruz.



2- Yönetim panelinden ise webhooks sekmesine gidiyoruz. Ve yeni bir webhooks oluşturmak için aşağıdaki butona tıklıyoruz.



3-Karşımıza yapabileceğimiz bir takım ayarlar karşımıza geliyor.

1.Tanımlamalar: Kullanmak istemediğimiz tanımlamaları kaldırabiliriz. Kesinlikle kullanmak gerekli olanları * işareti koyuldu.

Bunlar aslında yapmış olduğumuz sistemde bir yerlerin tetiklenmesini istiyorsak, bu değerleri kullanacağız.

-Async.join: Yalnızca IsPersistent true ise çağırılır. Arkadaşlar ile oynadığımız zaman sunucuda eşleştirme almayabiliriz yani sunucuda hata olabilir. Eğer oda bilgilerimizi web hizmetimizde tutuyorsak, oradan çekerek oyuncular oyunlarına kaldığı yerden devam etmesini sağlıyor.

-BaseUrl*: Hizmet vereceğimiz url burada belirtiyor olmamız gerekiyor. Web sitemizin kök dizin yolunu veriyoruz. (Klösör açmak istemiyorsan örneğin cancelik.com olarak da verebilirsin. Eğer klasör açmak istiyorsan ya da iç içe klasörler oluşturmak istiyorsan örneğin cancelik.com/Webrpc/queries sorgunun bulunduğu php dosyasının yolunu da verebilirsin.)

!Php dosasının bulunduğu klasörde asla ve asla / olmamalı. Dikkat

http://olcayegitim.com/webrpc → http://cancelik.com/webrpcfolder

-CustomHttpHeaders: Biz bir sorgu gönderdiğimiz zaman bir http başlığı gönderiyoruz aslında bu sorgunun nerden geldiğini ya da herhangi bir güvenlik yapısı oluşturmak istiyorsak onları belirttiğimiz alandır.

!Json formatında göndermemiz gerekiyor. Örnek;

{'X-secret':'Can'}

-HasErrorInfo: Eğer biz bunu true olarak belirtirsek, ve bir rpc gönderirsek ve buradan başarısız olursak burası çağırılır.

-IsPersistent: Bir oda oluşturulduğunda ya da kaldırıldığında geri bilgidirim almak için burası true olması gerekiyor.

-PathClose: Eğer sistemden bir oda kaldırılırsa burası tetiklenmektedir.

-PathCreate: Eğer ki yeni bir oda kurulursa bu durumu yakalayabiliriz. Burada kullanılan etiketler gerçekten çok önemli

-PathEvent: İslemci odada birere check pointler mevcut bayraklar eğer böyle olaylara ait işlemler yapmak istersek çağırılabilen bölüm.

-PathGameProperties: Herhangi bir oyuncu özelliği vs. değiştiğinde burayı çağırabilirsin.

-PathJoin: Bir oyuncu bir odaya katıldığında burası çağırılır.

-PathLeave: Bir oyuncu bir odadan ayrıldığında burası çağırılır.

6)WEBHOOKS & WEBRPC | Verilerin hazırlanması ve gönderilmesi | Cevap alınması

1-Bir üstteki ayarlarımızı kayıt ediyoruz ve kanca oluşturma işlemini tamamlamış oluyoruz. Şimdi ise veri hazırlama nasıl oluyor veri nasıl gidiyor ve cevaplar nasıl alınıyor o yapıları inceleyelim.

PHP:

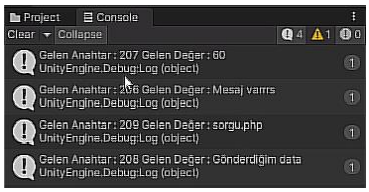
```
/*Php de photona cevap olarak gönderebileceğim
3 parametre bulunmaktadır.
1- Sonuç kodu
2- Mesaj
3- Data*/
```

Cevap:

Return Code bize 0 olarak dönüyorsa işlem başarılı demektir.

Değerleri karşılama:

207 → Result Code | 206 → Message | 209 → URL | 208 → Data

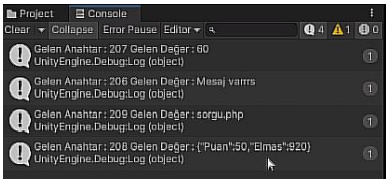


7)WEBHOOKS & WEBRPC | Gönderilen verileri okuma ve sonuca göre cevap gönderme

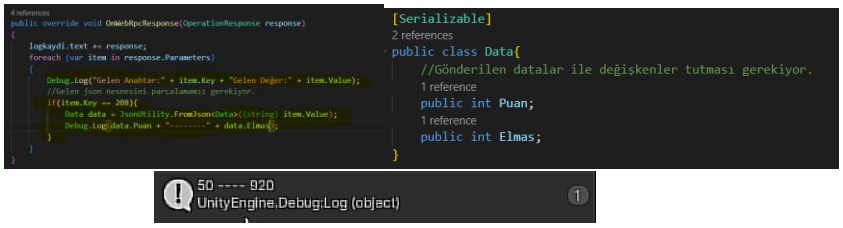
1-Verilerimizi gönderdik, karşıladık cevabımızı aldık. Şimdi ise gelen verileri okuma ve okuduğumuz sonuca göre cevap gönderme işlemlerine bakalım.

1.İlk olarak gelen json verisini parçalama ve okuma işlemi gerçekleştirelim.

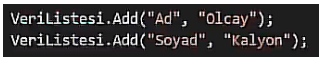
Okuma:

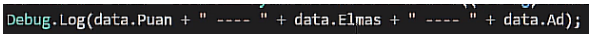


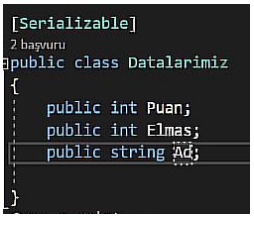
Parçalama



2. Şimdi gönderilen değerleri okuyup, gelen değerlere göre sorgulama yapıp veri gönderme işlemi yapalım.







3.Örneğin ID gönderdiğimizi düşündüğümüz bir yapı kullanalım. Yani göndermiş olduğumuz değere göre sorgulama yapalım.

```
VeriListesi.Add("ID", 10);
```

```
switch ($islem):

case "1":


$input = file_get_contents("php://input");
$sonuc=json_decode($input);
/* $sonuc->Ad; $sonuc->Soy...


  $veriler = new StdClass();
switch($sonuc->ID) :

 case "10":
    $veriler->Puan=150;
    $veriler->Elmas=120;

 break;
 case "665980":
    $veriler->Puan=950;
    $veriler->Elmas=40;

 break;

endswitch;


    $jsonveriler= json_encode($veriler);
```

```
150 ---- 120
UnityEngine.Debug:Log (object)
```

8)WEBHOOKS & WEBRPC | Oturum açarak oyuncunun verilerini veriabanından çekme

Şimdiki yapacağımız ise kullanıcı adı ve şifre girilerek veri tabanına bilgi göndereceğiz. Ve ilgili oyuncunun mevcut değerlerini alacağız.

1- Kullanıcı adı ve şifremizi inputfield dan alıyoruz.

```
0 başvuru
public void Baglan()
{

    PhotonNetwork.JoinLobby();


    AuthenticationValues kimlik = new AuthenticationValues(KullaniciAdi.text);
    PhotonNetwork.AuthValues = kimlik;

    VeriListesi.Add("AppId", "5353535345");
    VeriListesi.Add("AppVersion", "1");
    VeriListesi.Add("Region", "EU");
    VeriListesi.Add("UserId", PhotonNetwork.AuthValues.UserId);



    VeriListesi.Add("kullaniciadi", KullaniciAdi.text);
    VeriListesi.Add("sifre", Sifre.text);
    // 665980

}
```

2-Bakiye sorgulama isminde php dosyasında bir bölüm oluşturuyoruz. Direkt o bölümdeki verileri gidip alması için.

```
PhotonNetwork.WebRpc("sorgu.php?islem=BakiyeSorgula",VeriListesi,true);
```

```
case "BakiyeSorgula":

break;
```

3-Veri tabanı bağlantı işlemlerini php dosyası içerisinde gerçekleştiriyoruz.

```
try  {$db = new PDO("mysql:host...} catch (PDOException $e)
  {
  die($e->getMessege());
  }
```

4-Veri tabanımızı oluşturuyoruz. Bu sorgular ile veri ekleyebilir, güncelleyebilir, veri alabilir, veri silebilriz.

5-Gerekli kodları veri tabanından çekme işlemi vs. yaptıktan sonra rahatlıkça işlemlerimizi ayarlayabiliriz. Örnek php kodları:

sorgu.php

9)PHOTON STREAM YÖNETİMİ | Oyuncuların birbirlerine veri göndermesi

Bu bölümdeki son konumuz ise, Photon stream yöntemini inceleyeceğiz. Bu yöntem oyuncuların birbirlerine veri gönderip almasını sağlayan bir yapıdır.

Bu yöntemin amacı oyun içerisinde eğer bir oyuncu kendi ile ilgili bir veriyi ya da bir datayı oyun içerisindeki diğer kullanıcılara aktarmak istiyorsa stream yöntemini kullanabiliriz.

Peki! bu işlemi nasıl gerçekleştireceğim.

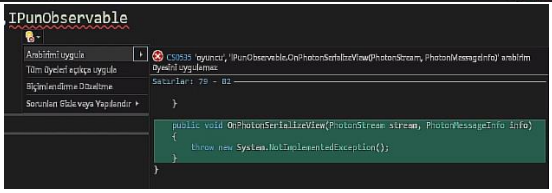
1-Biz oyuncumuzun transform, fiziksel vb. bilgilerini gerekli komponentler ile sağlayıp photon view komponenti içerisinde veriyoruz. Böylece diğer oyunculara verileri aktarmayı sağlıyor.

İşte burada bu yöntemler kesinlikle kullanabilir. Ancak stream yöntemi kullanarak da verileri aktarmayı gerçekleştirebiliriz.

Bu işlemi diğer komponentleri kapatıp, oyuncuya ait script dosyasını photonview içerisine verebilir ve bu şekilde de aktarabiliriz.

1-İlk olarak stream yapısını kullanabilmek istiyorsak, oyuncuya ait script dosyasında bir implement işlemi gerçekleştirmemiz gerekiyor.

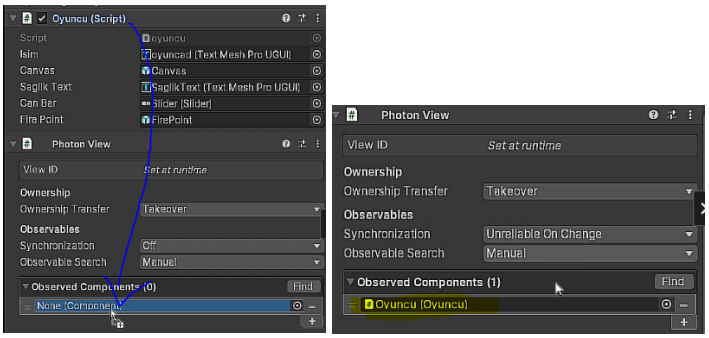
```
public class oyuncu : MonoBehaviourPunCallbacks,IPunObservable
```



2-Artık aşağıdaki fonksiyon ile veri gönderme işlemlerini gerçekleştireceğiz.

```
public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    throw new System.NotImplementedException();
}
```

3-Sonrasında ise script dosyamızı PhotonView komponenti içerisine gönderiyoruz.



4-Şim ise veri alma ve veri gönderme işlemlerine bakalım.

!stream.IsWriting: Yazan biziz | stream.IsReading: Başkası gönderiyor okuyan biziz.

```
10 başvuru
public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    if (stream.IsWriting)
    {
        stream.SendNext(transform.position);
        stream.SendNext(transform.rotation);

    }
    else if (stream.IsReading)
    {
        transform.position = (Vector3)stream.ReceiveNext();
        transform.rotation = (Quaternion)stream.ReceiveNext();

    }
}
```

5- Transform ve rotasyonu komponentler ile sağlıklı bir şekilde aktarabiliyoruz zaten. Şimdi ise biz veri aktarımı nasıl gerçekleşecek ona bakalım.

Birbirleri arasındaki data işlemlerini oyuncu scripti içerisinde yapabileceğin gibi farklı bir script dosyasından da yapabilirsin.

!Dataları gönderirken birden fazla ve farklı türlerde gönderebilirsin. Karşılarken de aynı gönderilen sırada alman gerekiyor.

```
public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    if (stream.IsWriting)
    {
        stream.Serialize(ref gidenveri);

        /* stream.SendNext(transform.position);
        stream.SendNext(transform.rotation);*/

    }
    else if (stream.IsReading)
    {
        gelenveri = (int)stream.ReceiveNext();
        /*  transform.position = (Vector3)stream.ReceiveNext();
          transform.rotation = (Quaternion)stream.ReceiveNext();*/

        GelendegeriYaz.text = gelenveri.ToString();
    }
}
```

```
if (Input.GetKeyDown(KeyCode.U))
{

    gidenveri += 4;
}
```

6-Stream yapısını farklı bir script dosyasında oluşturarak oyuncu içerisine atıp, sonra photonview içerisine atarak da rahatça kullanılabilir.

stream.cs