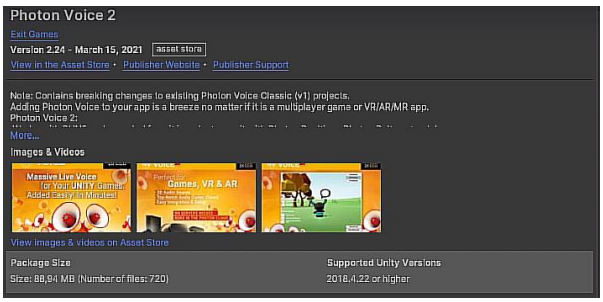
Voice Sistemi | Sesli İletişim

1)Voice paketi yükleme | Genel işleyişi anlatım | İlk testin yapılması

Bu bölümlerimizde ses sistemini inceleyeceğiz. Bu sistemimiz hem karışık hem de çok detaylı olduğunu söyleyebilirim.

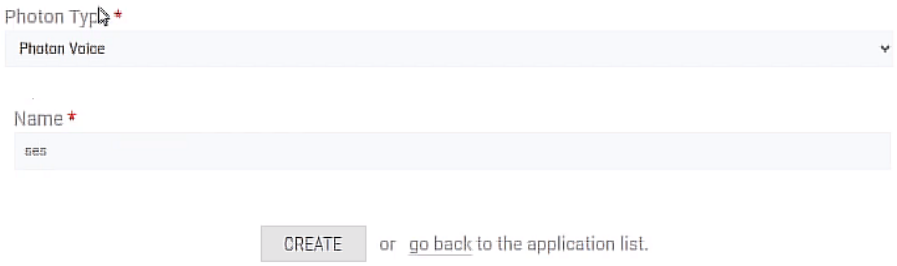
!Bu paketimiz Photon Pun ve Photon Chat içeren bir paket. Eğer projende Photon var ise o projenin üzerine yükleme yapılması tavsiye edilmemekte. Çünkü iki paketi de içerdiği için oyunuda voice sistemini kullanmak istiyorsan sadece bu paketi yüklemen yeterli olacaktır.

1-Photon Voice 2 paketini yüklüyoruz.

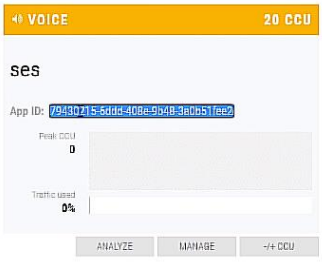


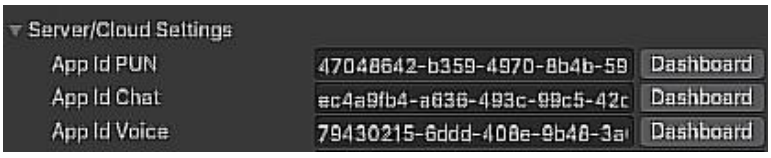
2-Yükleme işleminden sonra photon ayarlarından gerekli apilerin girilebileceği yerleri görüyoruz.

Photon içerisinden ses uygulaması oluşturma



!Aldığımız apiyi yine gerekli alana yapıştırıyoruz.





3-Photon Voice genel işleyiş.

→Ses iletişiminin sağlanabilmesi için olmazsa olmaz 3 adet component vardır.

Bunlar;

1.VoiceConnection: Bağlantıyı sağlayan ana componenttir.

2.Recorder: Ses akışını sağlayan componenttir.

3.Speaker: Gelen sesin oynatılmasını sağlar.

Yukarıdaki 3 component bir araya gelerek seslerin aktarımını ve bize gelen sesleri duyabilmemizi sağlar. Yani ses iletişiminin her aşamasını bu 3 component sağlar.

!Ses iletişimi yalnızca odlar içerisinden mümkündür. Lobby de ses iletişimi yapamazsın.

→Recorder componentinin 4 temel işlem aşaması vardır.

Bunlar;

1.Ses kaynağı girişi: Sesin ya da neyin aktarılacağını belirttiğimiz aşamadır. Photon bize Mikrofondan çıkan ses, ses klibi, kendi oluşturmuş oluşturduğumuz harici bir ses kaynağını biz aktarabiliriz.

2.Ses kalitesi: Nasıl yayınlayacağımızı belirttiğimiz alandır.

3.İletişim geçişleri: Konuşma süreklidir ya da bas konuş sistemi iletişimi sağlar. Nasıl bir iletişim geçişini sağlayacağımızı belirtiriz.

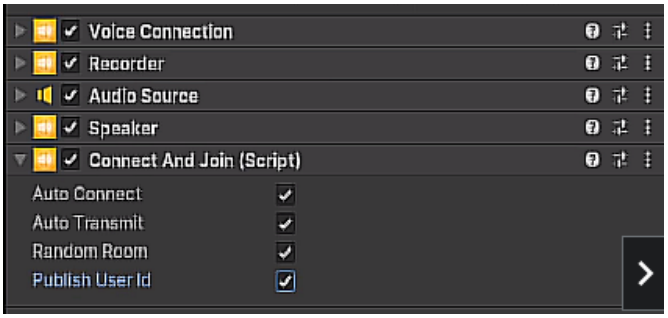
4.Alıcı hedefleri: Sesimizi kime ya da kimlere ileteceğimizi belirtebiliriz. Oyuncular kanallara göre konuşabilme ve dinleyebilme imkanı sağlar.

3-İlk testin yapılması (TEST)

1.Sahnemize boş bir obje oluşturuyoruz.

2.Voice Connection, Recorder, Speaker componentlerini ekliyoruz.

3.Uygulamayı başlattığımızda otomatik olarak bağlanabilmek için ise, Connect and join componentini ekliyoruz.



4.Speaker componenti yerine photon bize sunmuş olduğu speaker prefab objesini kullanabiliriz.

5. Voice Connection komponenti içerisine Recorder componentini ve Photonun bize sunmuş olduğu Speaker prefab objesini veriyoruz.

6.Voice Connection içerisine Photondan aldığımız ses app id vermemiz gerekmekte.

7.Ses girişimizin türünü Recorder içerisinde belirteceğiz. (Recorder Com.)

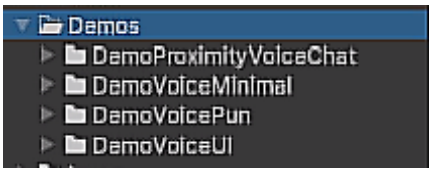
8.Oyun başlar başlamaz sesimizin nasıl aktarılacağını belirtmemiz gerekiyor. (Recorder Com.)

!Eğer bas konuş yapacak isek aşağıdaki seçeneği kapatmamız gerekiyor. Bas konuş mantığı ise şu örneğin p tuşuna bastığımızda aşağıdaki değişkeni true yapıp ses iletimini sağlayacağız. P tuşundan elimizi çektiğimizde false yapıp ses iletişimini kapatacağız. Olay bu.

---

2)Voice Connection | Recorder | Connect and Join | Ayarların anlatımı

1-İlk olarak projemizde herhangi bir sorun oluşturmaması için ise Photon ile gelen demoları kaldırıyoruz.



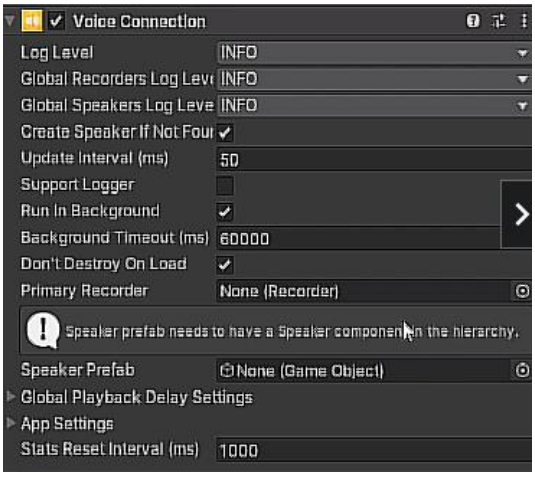
2-Sonrasında ise aşağıdaki package dosyasını projenize import etmelisiniz.

voiceBaslangic.unitypackage

3-Voice_Ana_Objemiz içerisine Voice Connection – Recorder – Connect and Join componentlerini ekliyoruz.

4-Bu bölümümüzde artık Component ayarlarına bakalım.

1-Voice Connection: Bağlantıyı sağlayan ana component.



1.LogLevel, Global Recorders LogLevel, Global Speakers LogLevel: Hangi durumlarda sana log kayıtlarını gösterme seçeneğini sunuyor. Genel olarak ERROR sunmasını istenir. Sadece uyarılar, bilgiler, her şeyi ve hiçbir şey seçeneği de bulunmakta.

2.Create Speaker or If Not Found: Eğerki oyuncuda speaker componenti yok ise otomatik oyun içerisinde oluştur seçeneğidir.

3.Update Interval (ms): Güncelleme sıklığıdır. Photon için default olarak gelen değerler daha öncesinden test edilmiş ve testler sonucunda belirlenen değerlerdir. Tabi ki değiştirmende herhangi bir sorun olmayacaktır.

4.Support Logger: Logların kayıt edilip edilmemesini belirttiğimiz alandır.

5.Run in Background: Bağlantımızın arka planda çalışıp çalışmayacağının kararını veriyoruz.

6.Background Timeout (ms): Bir önceki özellik ile paralel olan bir özelliktir. Arka planda ne kadar bekleyeceğini burada belirliyoruz. 60000ms (60s) eğer bu sürede arka planda oyun devam ederse süre bittikten sonra ses sistemini kapatacaktır.

7.Don’t Destroy On Load: Bağlantının kopmaması için sahne değişikliğinde kaybolmuyor.

8.Primary Recorder: Recorder verdiğimiz alandır.

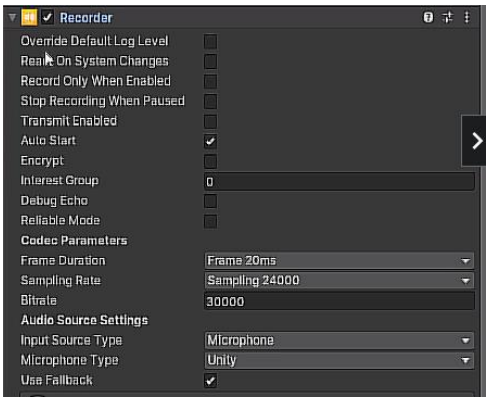
9.Speaker Prefab: Speaker verdiğimiz alandır.

10.Global Playback Delay Settings: Gelen oynatmanın gecikme değerleri. Kendimiz test ederek sesin ne kadar gecikeceğine kendimiz verebiliriz.

11.App Settings: Uygulamamızın ayarlarını yapabildiğimiz bölüm.

12.Start Reset Interval (ms): İstatistik aralığını buradan belirliyoruz. Örneğin 1 saniyede bir istatistikler sıfırlanacak, yeni istatistikler gelecek vs.

2-Recorder: Ses akışını sağlayan component.



1.Override Default LogLevel: Log kayıtlarının tutulmamasını istiyorsak seç…

2.React On System Changes: Sistem değişikliklerini yakalıyor.

-Skip Device Checks: Bir değişiklik olmasa bile sistemi sıfırlıyor.

3.Record Only When Enabled: Sadece component aktif iken kayıt yapıyoruz.

4.Stop Recording When Paused: Uygulama duraklatıldıktan sonra oyuna devam edersek tekrar ses kaydına giriyor

5.Transmit Enabled: Başlar başlamaz ses aktarımı aktifleştiririz.

6.Auto Start: Otomatik başlatma.

7.Encrpt: Sesler şifreli olarak gider.

8.Interest Group: Seslerimizi kime ileteceğimizi ve kimden alacağımızı belirttiğimiz alandır. (Kanal mantığı)

9.Debug Echo: Uygulamayı tek kişi test ediyorsa, tekrardan bize gelmesini sağlar.

10.Reliable Mode: Tüm giden ses akışları güvenilir mod da olacaktır.

Codec Parameters → Sesin kalitesini ayarladığımız ayalar.

12.Frame Duration: Giden ses akışı kodlayıcısı gecikmesidir.

13.Sampling Rate: Sesin saniyede kaç kez ölçüldüğünü belirtilen frekanstır.

14.Bitrale: Belirli saniye içerisinde işlenen veri (bit) sayısıdır.

Audio Source Settings → Ses kaynağı ayarları

15.Input Source Type: Giriş kaynağımızı belirliyoruz.

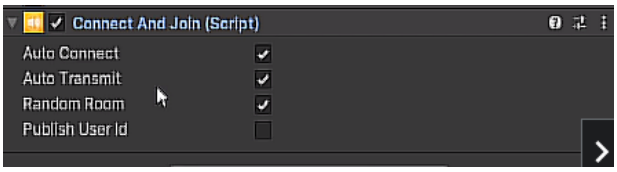
16.Microphone Type: Farklı apileri kullanarak mikrofonları seçmemizi sağlar.

17.Use Fallback: -

Voice Activity Deletction (VAD)

18.Delect: Gürültü engelleyicidir. (İstersek ince ayarda yapabiliriz)

3-Connect and join: Görevi bizi sisteme bağlamak ve nasıl bağlanacağına karar vermek.



1.Auto Connect: Otomatik bağlanmamızı sağlar.

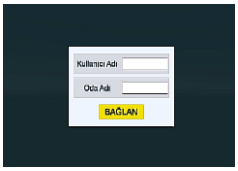
2.Auto Transmit: Otomatik sesin transferini sağlar.

3.Random Room: Random bir odaya girmek istersek bunu seçeriz. Eğer bu seçeneği kaldırırsak bizden bir oda ismi isteyecektir.

4.Publish User Id: Id paylaşıp paylaşmayacağımızı belirtiriz.

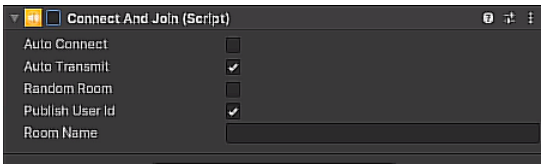
---

3)İstatistik bilgileri alma ve bağlantı kurma



1-Bu bölümümüzde bir oda adı ve kullanıcı adı girerek bağlanacağımız için, Auto Connect ve Random Room kapatıyoruz. User Id paylaşmasını istiyoruz.

İlk önce bu komponentimi kapatıyoruz. Bir odaya bağlanma işleminde bu componenti açıp, ses aktarım işlerini hazırlayacağız.



2-voicesistemi.cs (Package dosyası ile geliyor)

```
0 references
public void SohbeteBasla()
{
    /*Bağlanma butonuna tıklayınca*/
    if(string.IsNullOrEmpty(KullaniciAdi.text)){
        //Kullanıcı adı tanımsız olmamalı uyarı
        return;
    }
    KullaniciAdiValue = KullaniciAdi.text.Trim();
    voiceConnection.Client.LocalPlayer.NickName = KullaniciAdiValue;
    Kullanici_Adi_Text.text = KullaniciAdiValue;

    OdaAdiValue = OdaAdi.text.Trim();
    Oda_adi_Text.text  = OdaAdiValue;

    connectAndJoin.enabled = true;
    connectAndJoin.RoomName = OdaAdiValue;

    GirisPaneli.gameObject.SetActive(false);
    SohbetPaneli.gameObject.SetActive(true);

    if(!voiceConnection.Client.IsConnected){
        //Bağlantıda bir problem olursa tekrardan sisteme bağlıyoruz.
        voiceConnection.ConnectUsingSettings();
        GirisPaneli.gameObject.SetActive(false);
        SohbetPaneli.gameObject.SetActive(true);
    }
}
```

```
0 references
protected virtual void Update()
{
    //Panel aktifse sen bu kodu uygula diyoruz.
    if(SohbetPaneli.activeSelf){
        Baglanti_durumu_Text.text = voiceConnection.Client.State.ToString();
    }
}
```

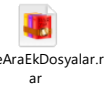
RemoteSpeakerUI.cs voicesistemi.cs

---

4)Mikrofonu sessize alabilme ve sesi açabilme işleminin yapılması

Bu bölümümüzde aşağıdaki toggle basıldığında sesin iletilmemesini sağlayacağız.

1-Bu işlemleri ister toggle ister bir tuşa bağlayarak işlemlerini yapabilirsin. İlk olarak aşağıdaki dosyayı projemize yüklüyoruz.

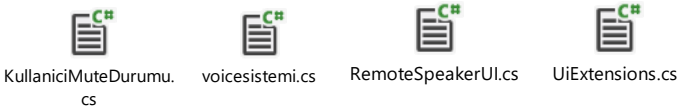


2-Sonrasında aşağıdaki script dosyalarının ne işe yaradıklarına bakalım.

→KullanıcıMuteDurumu.cs: Sessizde olup olmadığını ya da durumunu öğreneceğimiz dosyamız

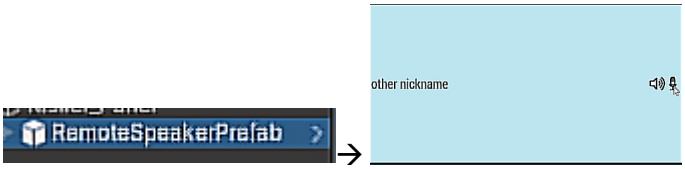
→UiExtensions.cs: Toggle tetikleme işlemini yapacağız.

→RemoteSpeaker: Kullanıcının sesine göre ikonları kontrol ediyor.



3- Script dosyamdan devam ediyorum.

Sohbette listelenecek kişileri hazırlıyoruz.



4- Script işlemleri (voicesistemi.cs)

```
0 references
void Start()
{
    connectAndJoin = GetComponent<ConnectAndJoin>();
    voiceConnection = GetComponent<VoiceConnection>();
   ToggleCallBack();
   ToggleValue();
}
1 reference
void ToggleCallBack(){
    //Toggle herhangi bir değişiminde bu fonksiyon tetiklenecek
    SesToggle.SetSingleOnValueChangedCallback(ToggleMute);
}
2 references
void ToggleValue(){
    //Oyuncunun mikrofon durumunu anlık bir şekilde set etmiş olacağız.
    SesToggle.SetValue(voiceConnection.Client.LocalPlayer.IsMuted());
}
1 reference
void ToggleMute(bool isOn){
    SesToggle.targetGraphic.enabled = !isOn;
    if(isOn)
        voiceConnection.Client.LocalPlayer.Mute();
    else
        voiceConnection.Client.LocalPlayer.Unmute();
}
```

```
0 references
protected virtual void Update()
{
    ToggleValue();
    //Panel aktifse sen bu kodu uygula diyoruz.
    if(SohbetPaneli.activeSelf){
        Baglanti_durumu_Text.text = voiceConnection.Client.State.ToString();
    }
}
```

```
1 reference
protected virtual void OnActorPropertiesChanged(Player targetPlayer, Hashtable changedProps)
{
   //Oyuncunun herhangi bir ayarı değişirse, burası çalışacak
   if(targetPlayer.IsLocal){
        bool IsMuted = targetPlayer.IsMuted();
        voiceConnection.PrimaryRecorder.TransmitEnabled = !IsMuted;
        SesToggle.SetValue(IsMuted);
   }

}
```

```
0 references
public void SohbeteBasla()
{

    /*Bağlanma butonuna tıklayınca*/
    if(string.IsNullOrEmpty(KullaniciAdi.text)){
        //Kullanıcı adı tanımsız olmamalı uyarı
        return;
    }
    KullaniciAdiValue = KullaniciAdi.text.Trim();
    voiceConnection.Client.LocalPlayer.NickName = KullaniciAdiValue;
    Kullanici_Adi_Text.text = KullaniciAdiValue;

    OdaAdiValue = OdaAdi.text.Trim();
    Oda_adi_Text.text  = OdaAdiValue;

    connectAndJoin.enabled = true;
    connectAndJoin.RoomName = OdaAdiValue;

    GirisPaneli.gameObject.SetActive(false);
    SohbetPaneli.gameObject.SetActive(true);

    if(!voiceConnection.Client.IsConnected){
        //Bağlantıda bir problem olursa tekrardan sisteme bağlıyoruz.
        voiceConnection.ConnectUsingSettings();
        GirisPaneli.gameObject.SetActive(false);
        SohbetPaneli.gameObject.SetActive(true);
    }
    //Callback başlatıyoruz.
    voiceConnection.Client.AddCallbackTarget(this);
}
```

```
0 references
private void OnDisable()
{
    //Dosya disable olduğunda callback kapatıyoruz.
    voiceConnection.Client.RemoveCallbackTarget(this);
}
```

---

5)Oyuncu Remote Speaker bilgisinin eklenmesi

Script dosyamızdan devam ediyoruz.

Speaker oluşturuyorum abone oluyorum ve callback çalıştırdım.

```
voiceConnection.SpeakerLinked += OnSpeakerCreadted;
voiceConnection.Client.AddCallbackTarget(this);
```

```
1 başvuru
protected virtual void OnSpeakerCreadted(Speaker speaker)
{
    speaker.gameObject.transform.SetParent(KisilerinPaneli, false);
    RemoteSpeakerUI remoteSpeakerUI = speaker.GetComponent<RemoteSpeakerUI>();
    remoteSpeakerUI.Init(voiceConnection);
    speaker.OnRemoteVoiceRemoveAction += OnRemoteVoiceRemove;

}

1 başvuru
private void OnRemoteVoiceRemove(Speaker speaker)
{
    if (speaker != null)
        Destroy(speaker.gameObject);
}
```

Aboneliği kaldırıyorum.

```
Unity İletisi | 0 başvuru
private void OnDisable()
{
    voiceConnection.SpeakerLinked -= OnSpeakerCreadted;
    voiceConnection.Client.RemoveCallbackTarget(this);
}
```
