



DESIGN IT! - Proje Final Raporu

TEAM DESIGN

Murat Şahin - 191101010

Mehmet Can Şakiroğlu - 191101001

Gökçe Başak Demirok - 191401005

Barış Utku Mutan - 201104082

BİLGİSAYAR / YAPAY ZEKA MÜHENDİSLİĞİ BÖLÜMÜ

TÜRKİYE ODALAR VE BORSALAR BİRLİĞİ EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

ANKARA

4 MAYIS 2023

İÇİNDEKİLER

1. Projenin Amacı	3
2. Hedef Kullanıcı Kitlesi	3
3. Kullanıcı Senaryoları	3
4. Uygulama Fonksiyonlarının Tanıtımı	4
4.1. Kimlik Doğrulama	4
4.2. Kullanıcı Odasının - Eşyalarının Taranması ve Erişimmesi	8
4.3. Eşyaların Türlerine Ait Etiketlerinin Belirlenmesi	10
4.3.1. Object Detection Destekli Otomatik Etiket Belirleme Yöntemi	10
4.3.2. Sesli Komut Yardımı ile Manuel Etiket Belirleme Yöntemi	13
4.3.3. Object Detection Destekli Etiket Belirleme Sistemi Kapsamında Kullanılan Object Detection Sistemi: YOLO	14
4.3.4. Eşyaların Türlerine Ait Etiketlerinin Belirlenmesi Sisteminin Diğer Sistemlerle Entegrasyonu ile İlgili Not	17
4.4. Oyun İçi Model Menüsü & Dinamik Model Yükleme	17
4.4.1. Yürütme zamanında model yükleme	17
4.4.2. Model menüsü	17
4.5. Obje Manipülasyon Sistemi	18
4.6. Sesli Komut Tanıma & Algılama Sistemi	20
4.6.1. Ev Eşyası İstemeliye Alaklı Komutlar	20
4.6.2. Menü ve Oyun Konulu Komutların Algılanması (Projeye entegre edilemedi)	20
4.6.3. Ev Eşyası Piyasa Araması İstemeliye Alaklı Komutlar (Projeye entegre edilemedi)	21
4.7. Inventory & Interact	21
4.7.1. Inventory	21
4.7.2. Interact	23
4.8. Video & Voice Chat	24
5. Sistem Mimarisi ve Sistem Unsurları Arası İlişkiler	25
6. Alt Sistemlerin Detayları	26
6.1. Doğal Dil İşleme - Sesli Komut	26
6.1.1. Speech Recognition (Ses verisini metne dönüştürme)	26
6.1.2. Command Detection (Metinden Aksiyona Geçme)	27
6.2. Ses ile tetiklenen yürütme sırasında yükleme	27
7. Takım İçi İş Bölümü	28
8. Referanslar	30

1. Projenin Amacı

Projemiz iç mimarlık alanında hem müsteri hem de mimar için verimi artırmaya yönelik olacak şekilde VR/AR gözlük kullanarak mimar ve müsterinin öngörü kazanmasını sağlar. Bunu yaparken de ayrıca kullanılan sistemlerle içinde bulunulan odanın özelliklerini de koruyarak çok etkili bir deneyim sunmayı hedefler.

2. Hedef Kullanıcı Kitlesi

Özellikle tasarım ve iç dizayn ile uğraşan insanların genel sorunlarından birisi yıkıp tekrar başlamaktır. Problem tanımında da bahsedildiği gibi dizayn süreci, istenilene erişilmediğinde kendini tekrarlayan bir yapıya sahiptir. En başta, ilk kararı verirken bile, o an oradayken hızlıca akılda oluşan dizaynı denemek, tasarımcı için yapacağını bulacak anlamına gelmese bile neyi yapmayıacağı konusunda önemli bir öngörü sağlayabilir.

Projemizin çıkış noktası iç mimarlardır ve ana hedef kitlesini onlar oluşturmaktadır. Ancak kendi evinin dizaynını değiştirmek ya da dizayn denemek isteyen ev sahipleri de uygulamamızı gönül rahatlığıyla kullanıp, “şuraya bir masa siğar mı?” sorusunun cevabını alabilir.

3. Kullanıcı Senaryoları

- Kullanıcı olarak uygulamaya giriş yaparken kimliğini kolayca doğrulayabilmek isterim ve bunun oturumlar arasında da korunmasını isterim.
- Kullanıcı olarak içinde bulunduğu odanın boyutlarını kolaylıkla ve büyük doğrulukla uygulamaya aktarıp sanal gerçeklikte aynı boyutlarda dizayn yapabilmeliyim. Böylelikle odayı birebir dizayn edebilirim ve gerçek hayatı ölçülerimi oturtabilirim.
- Kullanıcı olarak odada bulunan eşyaları kolaylıkla ve büyük doğrulukla uygulamaya aktarmanın bir yolunu isterim ve eşyaların türünün otomatik olarak tespit edilmesini isterim. Böylelikle o eşyaları da dizaynimda kullanabilirim ve odada güvenlice hareket edebilirim.
- Kullanıcı olarak dekorasyon işleminde aklımdaki eşyaların popüler olanlarının önüme fiyatlarıyla beraber sunulmasını isterim, bu sayede bütçeme uygun bir dekorasyon yapıp yapmadığımı anlayabilir, buna göre farklı bir dizayn yapmaya yönelebilirim.

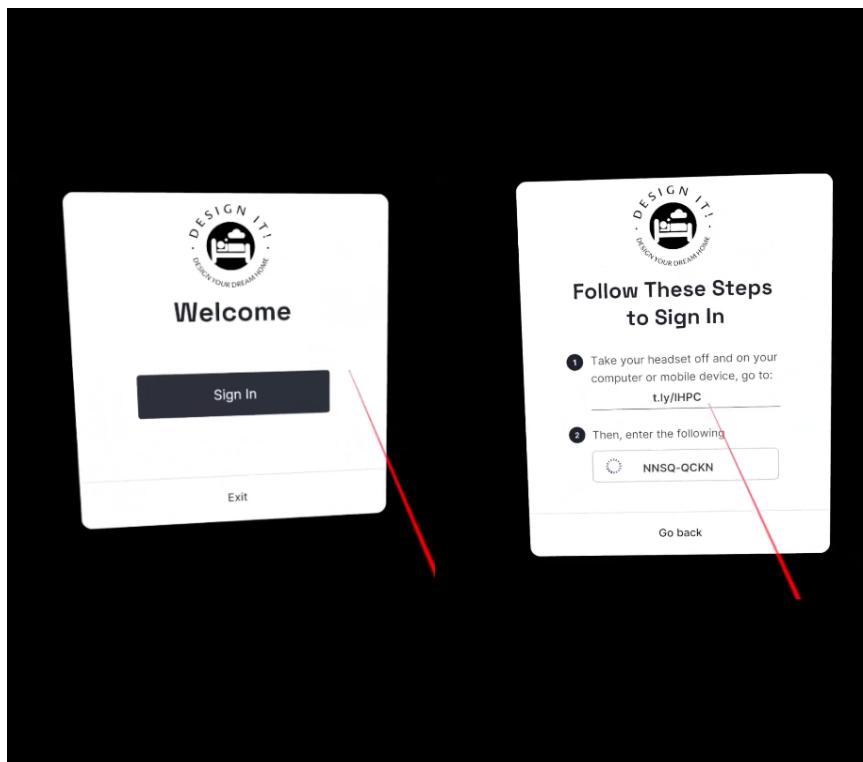
- Kullanıcı olarak, sesli komut aracılığıyla odaya yerleştirilmek istenilen yeni bir ev eşyasının söylemem sonucunda yerleştirilebilir modellerinin yanda bir eşya listesi menüsü olarak olmasını, örneğin sesli olarak “round table” dedığımde kullanılabılır çeşitli yuvarlak masa modellerinin yan menü halinde belirmesini isterim.
- Kullanıcı olarak, gerek daha önce odada olan eşyalar listesinden bir eşya modelini, gerekse odaya yeni bir eşya eklenmek istenildiğinde sesli komut özelliği sayesindeki arama sonrasında listelenen eşya listesi menüsündeki bir eşya modelini; çıkan listeden seçip, tutup sürükleyerek odanın istediğim yerine bırakabilmeyi isterim.
- Kullanıcı olarak, odadaki bir eşyanın tut-sürükle-bırak metodu ile yerinin değiştirilebilmesi ve döndürülebilmesi sayesinde dizayn sürecinde eşyalarının yerinin ve rotasyonun düzenlenenebilmesini isterim.
- Kullanıcı olarak kullanım esnasında istediğim doğrultusunda başka birine, örneğin bir iç mimar isem müşterime, canlı olarak dizaynı gösterebilmek isterim.

4. Uygulama Fonksiyonlarının Tanıtımı

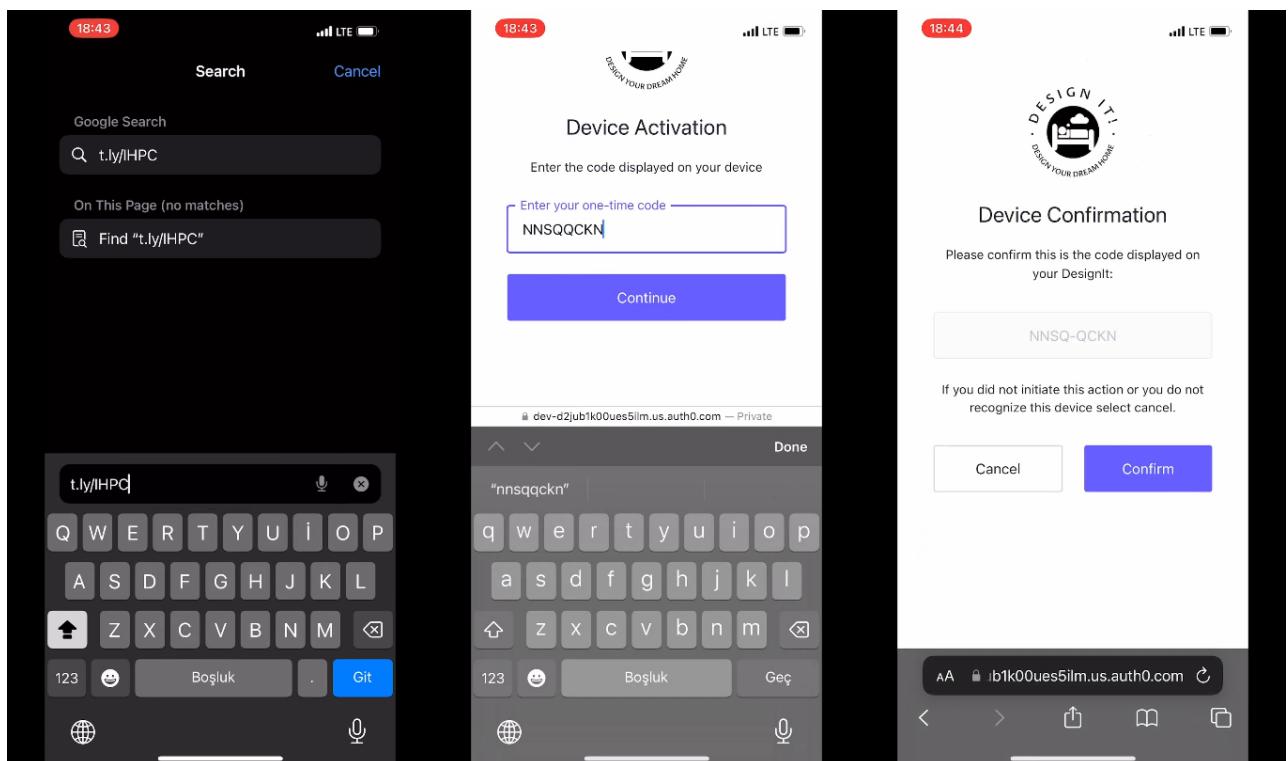
4.1. Kimlik Doğrulama

Alışlagelmiş sistemlerde kullanıcılar kullanıcı adı ve şifrelerini VR kontrolcülerile girmeye çalışıyorlar ve bu büyük bir zorluğa yol açıyor. Biz uygulamamızda kullanıcıların kimliğini doğrulamak için alışılmışın dışında bir çözüm tercih ettik. [Auth0 Lab](#) [1] tarafından sunulan Unity SDK yardımıyla kullanıcıların kimliklerini doğrulayabilmesini sağladık.

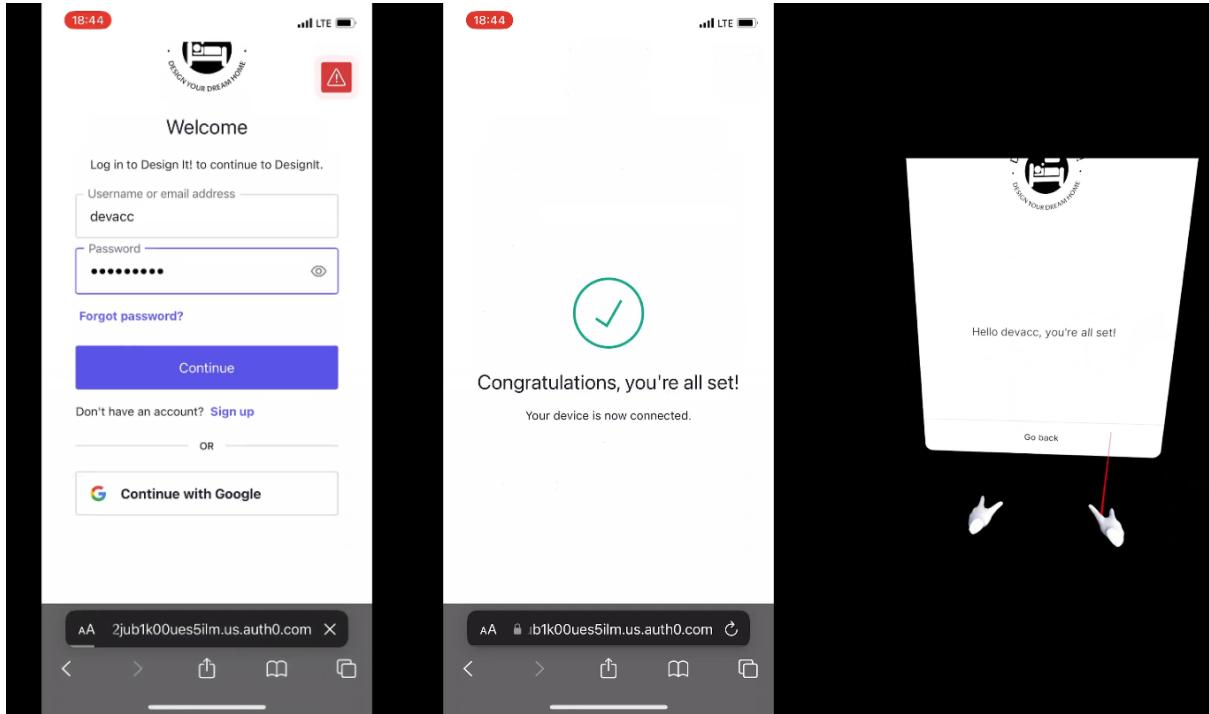
Bu sistemde kullanıcılar uygulamamızı açtıktan sonralarına 8 haneli bir kod ve kısa bir adres geliyor ve yakınlarındaki bir cihazdan **bu adresi açmaları** isteniyor.



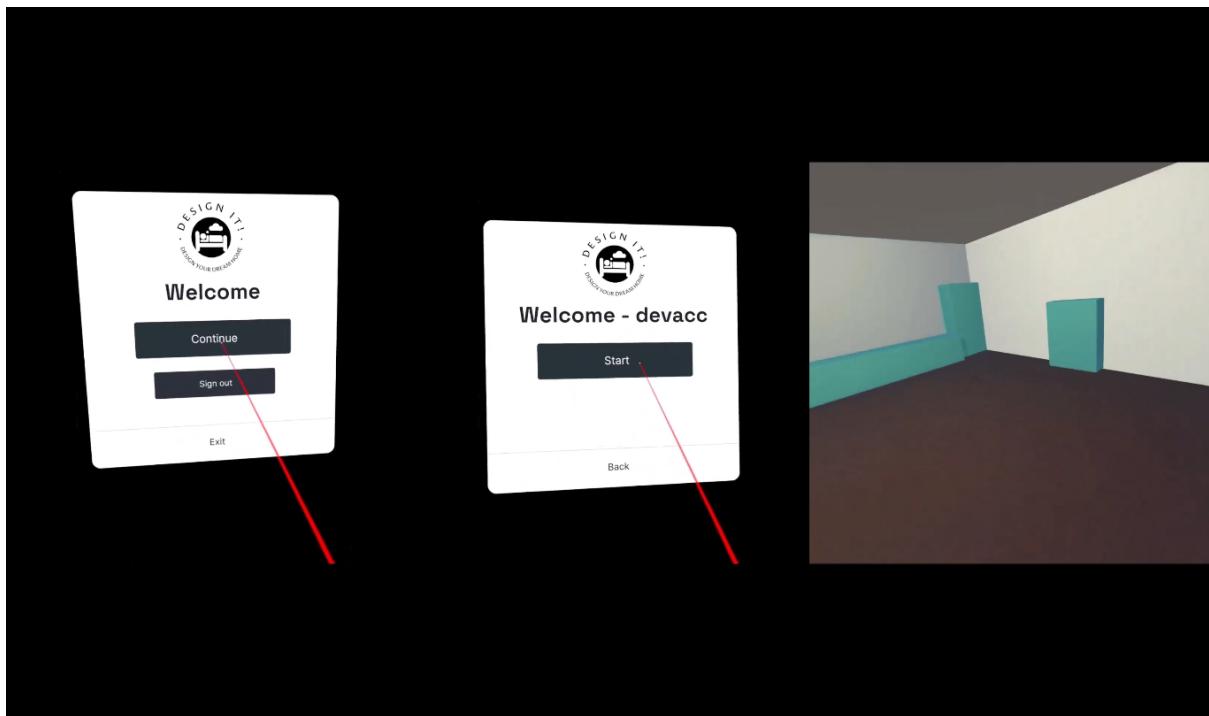
Adresi açtıktan sonra karşısına çıkan bölüme VR gözlüğünde gösterilen olan bu **kodu giriyor.**



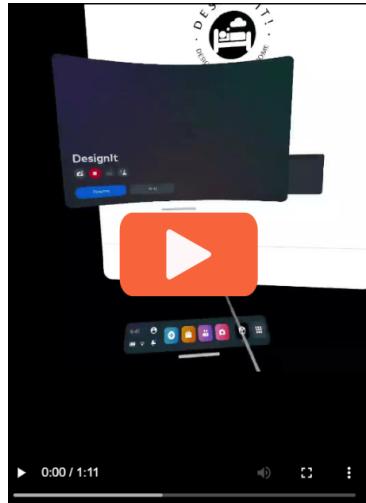
Daha sonra kullanıcı kullanıcı adı ve şifresiyle mobil cihazından **kolayca giriş yapıyor** ve senkronize bir biçimde **VR'da da giriş yapmış oluyor**.



Kullanıcımız kimliğini başarıyla doğruladıktan sonra uygulamamızda devam edebiliyor.



Tüm bu işlemlerin adım adım yapıldığı demo videomuza aşağıdan erişebilirsiniz:



(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

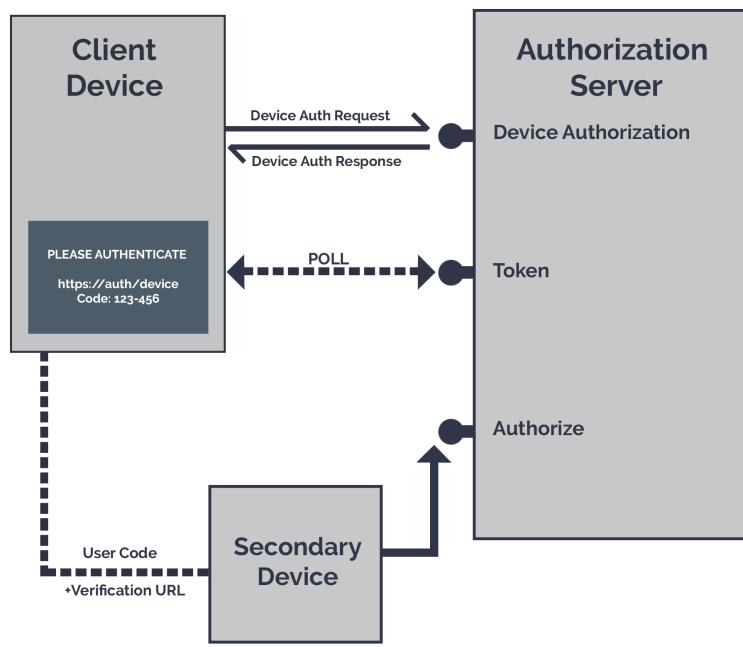
Kullanıcılar hakkında bilgilerin tutulduğu ve sistem hakkında çeşitli düzenlemelerin yapılabileceği Auth0 panelini aşağıdaki görselde görebilirsiniz.

Name	Connection	Logins	Latest Login
devacc@designit.com	Username-Password-Authenti...	8	14 days ago
muratsahin@test.com	Username-Password-Authenti...	0	never
cansakiroglu@test.com	Username-Password-Authenti...	0	never
bdemirok@test.com	Username-Password-Authenti...	0	never
barismutan@test.com	Username-Password-Authenti...	0	never

Implementasyon Detayları

Auth0 tarafından en son 2022 başlarında güncellenmiş ve aslında açık bir şekilde “Proof Of Concept” olduğu belirtilen bir GitHub reposunu kullandık. Projemize ekledik ve eski olduğu için çıkan sorunları düzelttik, güncelledik. Auth0 tarafında kendi hesabımızı açtık ve gerekli şekilde ayarlamaları yaptıktan sonra Unity ile bağlantısını sağladık.

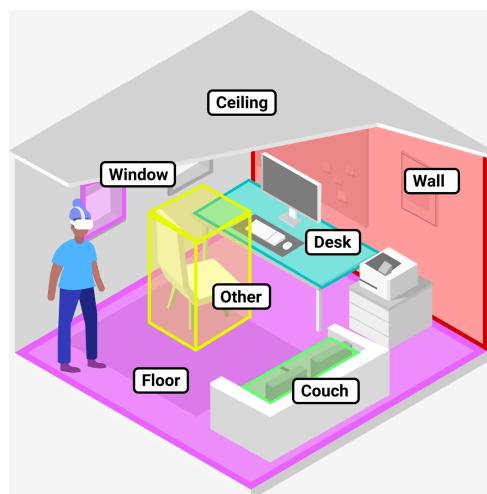
Kullanılan sistemin yapısı aşağıdaki şemadan da görülebilir.



4.2. Kullanıcı Odasının - Eşyalarının Taranması ve Erişimmesi

Kullanıcının bulunduğu oda hakkındaki bilgilerin (boyut, duvar konumu, eşya konumları) çekilebilmesi taskını gerçekleştirebilmek için RoomMapper eklentisinin kullanılabileceğini düşünmüştük. Fakat daha sonradan Oculus'un [Presence platform](#) [2] adı altında duyurduğu gerçek ve sanal dünyayı birleştirmeyi hedefleyen bir oluşumu gördük. Bu hedefe yönelik de [Scene Capture](#) [3] denilen bir sistem oluşturmuşlar.

Bu sistem sayesinde kullanıcı odasını işletim sistemi düzeyinde tanıtabiliyor, biz de bu bilgiyi Oculus tarafından sunulan yardımcı bileşenler yardımıyla çekebiliyoruz. Böylelikle odanın ve eşyaların seçilmesi işlevi daha basit bir hale geliyor. Bu konu hakkında örnek bir görseli aşağıda görebilirsiniz.



Bu sistemi gördükten sonra, RoomMapper eklentisini tamamen bir kenara bırakıp bu eklentiyi araştırmaya odaklandık ve çok iyi sonuçlara ulaştık. Bu sistem kullanıcının oda bilgisini OS düzeyinde headset'e aktarabilmesini sağlıyor. Uygulama geliştiriciler bu bilgileri API'lar aracılığıyla kullanabiliyor. Sistemin internette yeterli bir dokümantasyonu maalesef bulunmamakta, kaynak kodu kendimiz okuyarak çoğu şeyi anlayabildik.

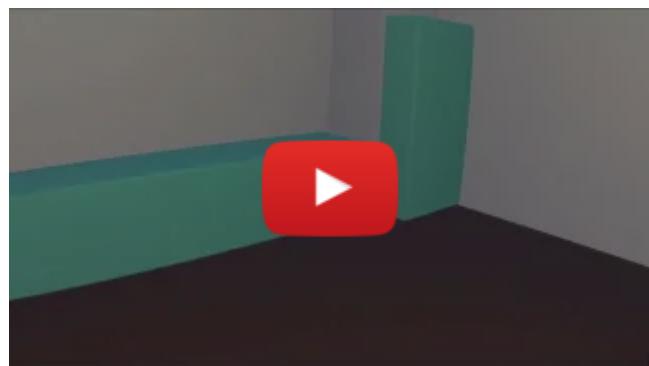
Böylelikle kullanıcımız uygulamaya girdiğinde eğer daha önceden odasını tanıtmamışsa tanıtması için bir diyaloga yönlendiriliyor, kullanıcı öncelikle odasının duvarlarını ve daha sonra da eşyalarının sınırlarını seçiyor. Daha sonra ise uygulamamızda odası tanımlı bir şekilde devam ediyor, odasının içinde gerçek hayatı yürüken birebir ölçekli bir şekilde uygulamamızda da hareket edebiliyor.

Kullanıcının odasını tanıttığı demo videosuna aşağıdan ulaşabilirsiniz:



(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

Kullanıcının odasına uygulamamızdan eriştiğimiz demo videosuna aşağıdan ulaşabilirsiniz:



(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

Implementasyon Detayları

Oculus Integration paketi sayesinde Unity'e eklenen *OVRCameraManager* bileşeni genel olarak bu işlevleri yönetiyor. Uygulamamıza giriş yapıldığında otomatik olarak *OVRCameraManager.RequestSceneCapture()* metodu çağrılarak kullanıcıya odasını tanıtması talebi iletiliyor. Daha sonra kullanıcından gelen oda bilgisi de

OVRSceneManager.LoadSceneModel() metodu çağrılarak uygulamamıza çekilebiliyor. Bilgiler Unity'e çekildikten sonra yazdığımız scriptler ile bunları kullanabiliyoruz.

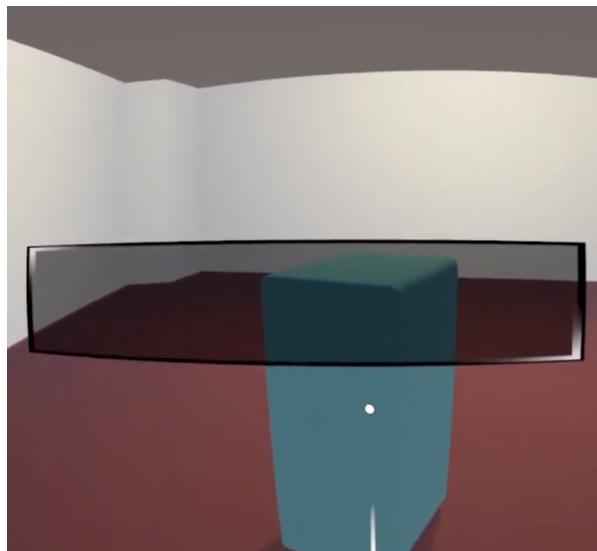
4.3. Eşyaların Türlerine Ait Etiketlerinin Belirlenmesi

Eşyaların türlerinin belirlenmesi, eşyalara ait Unity'de kullanılabilen modelin aratılabilmesi için gereklidir. Çünkü, odada halihazırda mevcut olan eşyaların da modellerinin odada gösterilebilmesi için bu eşyaya ait modellerin, modeller içerisinde aranmasını sağlamak amacıyla bir etikete ihtiyaç duyulmaktadır. Bu sebepler dolayısıyla odadaki eşyalara ait etiketin belirlenmesi üzerine çalıştık ve kullanıcının seçtiği yöntemle odadaki bir eşyasına ait etiketi belirleyebilmesini sağladık.

Kullanıcıya, odada var olan bir eşyasına etiket belirleme işlemini gerçekleştirebilmesi için iki farklı yöntem sunuyoruz. İlk kullanıcıının konuşmayı tercih etmediği bir ortamda olması veya çok fazla etiketleyeceği eşya olması durumunda faydalı olacağını düşündüğümüz Object Detection destekli etiket belirleme yöntemi. İkincisi ise kullanıcının etiketleyeceği az sayıda eşya olması, uygulamayı kullandığı esnada konuşması için bir engel teşkil eden bir durum olmaması durumunda kullanımının tercih edilmesi daha mantıklı olacak olan Sesli Komut ile etiket belirleme yöntemi. Bu iki yöntemimiz ve uygulama içerisindeki kullanımları hakkında detaylı bilgiler aşağıdaki alt başlıklar içerisinde paylaşılmıştır.

4.3.1. Object Detection Destekli Otomatik Etiket Belirleme Yöntemi

Kullanıcı bir eşyanın türünün ne olduğuna dair etiketin Object Detection destekli etiket belirleme yöntemi yardımıyla otomatik olarak algılanmasını isterse, sol controller'ı ile eşyayı point ederken sol controller'ındaki Y tuşuna basıyor.



Bu noktada, cihazın passthrough özelliğine geçiliyor ve kullanıcının kendi odasına ve tabii asında, odasındaki eşyasına ait görüntü alınıyor.



Elde edilen görüntü ise server'a gönderiliyor. Server kısmında object detection modeli ile o görüntüdeki eşyaya ait etiket bulunuyor ve dönülüyor.

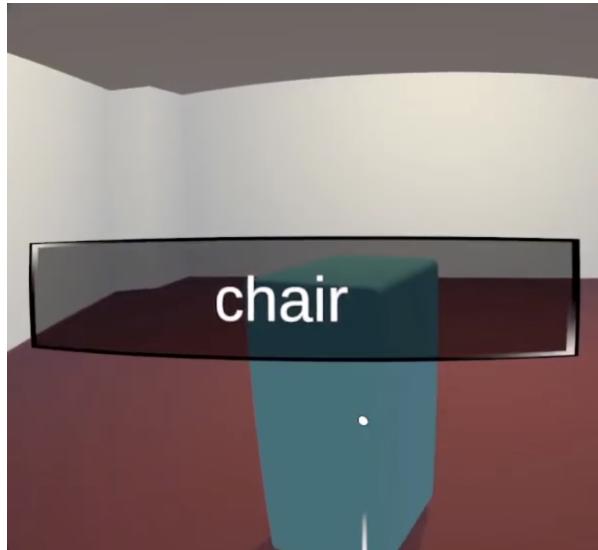
```
YOLOR 008b72b torch 2.0.0+cpu CPU

Fusing layers...
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
Model Summary: 306 layers, 36905341 parameters, 36905341 gradients
Convert model to Traced-model...
traced_script_module saved!
model is traced

chair
192.168.137.74 - - [18/Apr/2023 17:33:19] "GET / HTTP/1.1" 200 -
192.168.137.74 - - [18/Apr/2023 17:33:19] "GET / HTTP/1.1" 200 -
□
```

(Bu kısımda projemiz kapsamında kullanmış olduğumuz, bu yöntemin temelini oluşturan YOLO isimli real-time object detection algoritmasından ve projemizdeki kullanış biçimimizden [4.3.3. section](#)'ında bahsedilmiştir.)

Bu etiket ise o kullanıcının başta işaret etmiş olduğu eşya modeline atanıyor.



Bu şekilde kullanıcı, etiketini belirlemek istediği eşyanın türünü belirten etiketi, eşyaya, tek bir tuşa basma hareketi dışında bir şey yapmasına gerek kalmadan atamış oluyor.

Bu yöntemin temel dezavantajı ise, sonuç itibarı ile çalışmış olan Object Detection sisteminin en nihayetinde bir tahmin yapıyor olması ve her ne kadar yüksek doğrulukla tahmin yapabiliyor olursa olsun bir yanlışlık (yanlış etiket belirleme) yahut bulamama (etiket belirleyememe) ihtimalinin her zaman mevcut olması. Ancak bu gibi durumlarda, kullanıcımızın Sesli Komut Yardımı ile Etiket Belirleme Yöntemi ile o eşyaya ait etiketi manuel olarak tekrar belirleyebilmesi mümkün. Bu şekildeki bir durumun yaşanmadığı bir senaryoda ise kullanıcımız bu yöntem ile eşyasının etiketini belirleme işlemini tamamlamış oluyor.

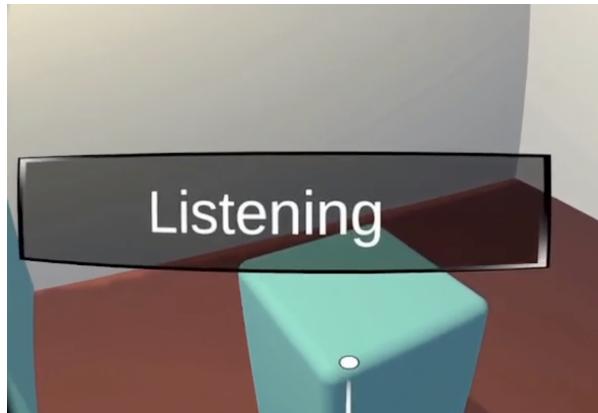
Bu yöntem ile ilgili geliştirmelerimizi göstermiş olduğumuz video şu şekilde: (Sağda VR headset'ten kaydedilen görüntü mevcutken, solda ise çözümümüzün işleyişini/akışını gösteren ilgili log çıktıları mevcuttur.)



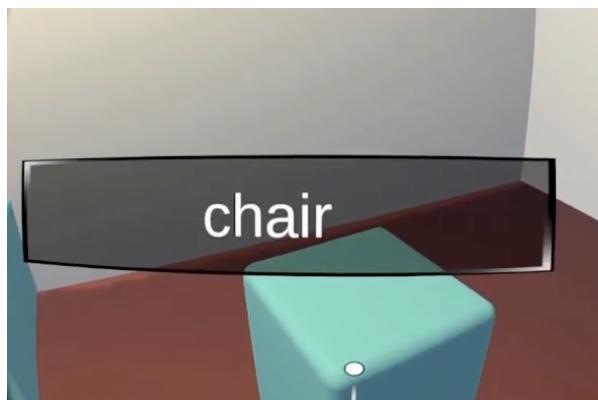
(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

4.3.2. Sesli Komut Yardımı ile Manuel Etiket Belirleme Yöntemi

Kullanıcı bir eşyanın türünün ne olduğuna dair etiketi kendisi manuel bir şekilde Sesli Komut yardımcı ile etiket belirleme yöntemini kullanarak belirlemek isterse, sol controller'ı ile eşyayı point ederken sol controller'ındaki X tuşuna basıyor. Bu noktada ise, kullanıcının diyecekleri dinlenilmeye başlanıyor.



Sonrasında voice transcript yapılarak kullanıcının söyleyerek belirtmiş olduğu etiket elde ediliyor.

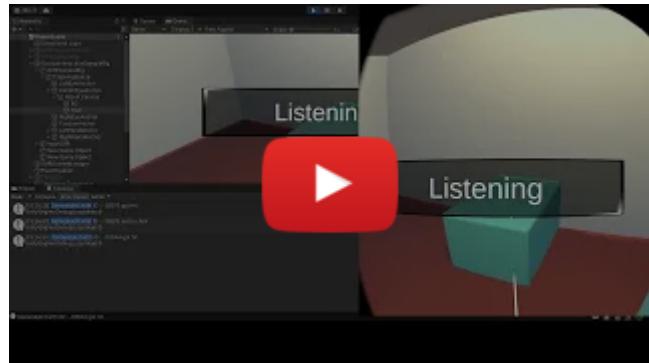


Bu etiket ise o kullanıcının başta işaret etmiş olduğu eşya modeline atanıyor.

```
! [19:26:28] Gameobject with ID : -18370 got hit.  
UnityEngine.Debug:Log (object)  
! [19:26:31] Gameobject with ID : -18370 set to chair  
UnityEngine.Debug:Log (object)
```

Bu şekilde kullanıcı, etiketini belirlemek istediği eşyanın türünü belirten etiketi, eşyaya başarıyla atamış oluyor.

Bu yöntem ile ilgili geliştirmelerimizi göstermiş olduğumuz video ise şu şekilde: (Sağda VR headset'ten kaydedilen görüntü mevcutken, solda ise çözümümüzün işleyişini/akışını gösteren ilgili log çıktıları mevcuttur.)



(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

4.3.3. Object Detection Destekli Etiket Belirleme Sistemi Kapsamında Kullanılan Object Detection Sistemi: YOLO

YOLO (You Only Look Once), ya da spesifik olarak bizim kullanmış olduğumuz [YOLOv7](#) [4] (You Only Look Once - version 7) bir nesne algılama ve sınıflandırma modelidir. Projemizde model, kullanıcının etiketinin belirlenmesini istediği bir eşyasının etiketinin otomatik olarak belirlenmesini sağlamak için kullanıldı. Çünkü modelin, kullanıcının işaret ettiği yerde bulmuş ve sınıflandırmış olduğu nesnenin etiketi aslında bizim elde etmek istediğimiz esya türü etiketine karşılık gelmekte.

Modelin projemiz açısından bir diğer avantajı ise halihazırda furniture içerisinde bol miktarda furniture class'ı bulunduran COCO dataseti ile eğitilmiş halinin genel kullanıma açık bir şekilde internetten ulaşılabilir olması. Bu sayede; veri seti oluşturma, eğitme gibi oldukça masraflı ve süreci zorlaştıracak adımları başarıyla atlayabilmemiz mümkün olmaktadır.

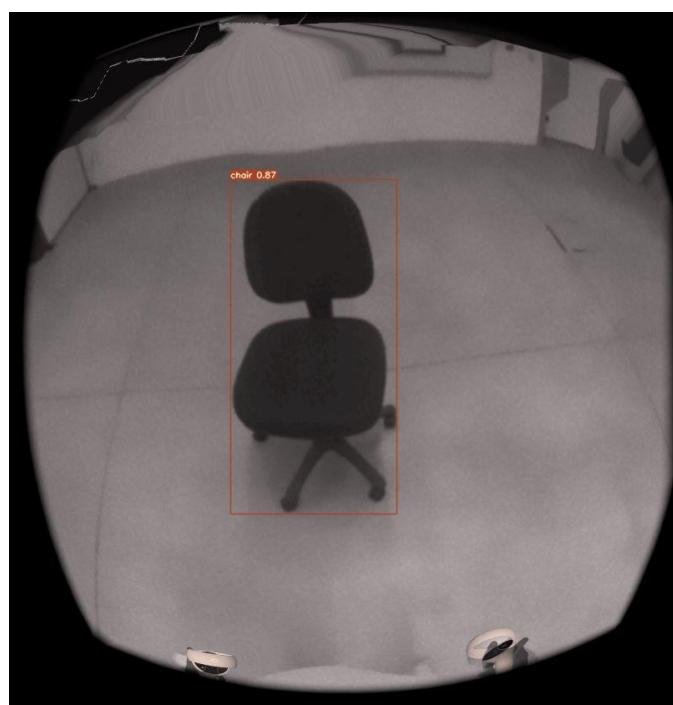
Kullanmış olduğumuz modeli tabii sadece furniture'lara karşı response üretecek şekilde güncelledik, burada bahsettiğimiz detect edilebilir furniture'ların listesi işe şu şekilde:

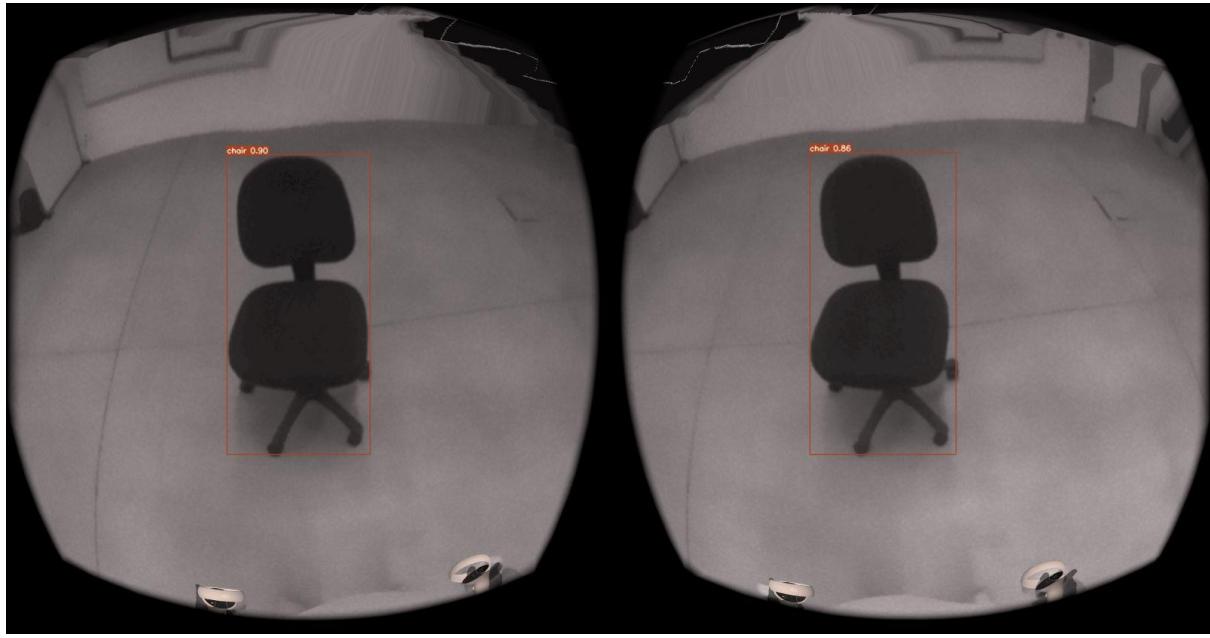
- 'chair'
- 'couch'
- 'potted plant'
- 'bed'
- 'dining table'
- 'toilet'
- 'tv'
- 'laptop'
- 'microwave'
- 'oven'
- 'sink'
- 'refrigerator'

- 'clock'
- 'vase'

Bir diğer tasarım kararımız ise, üzerinde çalıştığı image için birden fazla eşya bulma durumunda model içerisinde yine network'ün bir çıktısı olarak kararlaştırılan "confidence" değerini kullanarak hangi eşyayı seçeceğimiz kararını vermemiz. Bu yaklaşımın pratikte oldukça iyi çalıştığını deneyimledik. Dolayısıyla bu gibi durumlarda, model "confidence" en yüksek olan eşyayı dönmektedir, birden fazla eşya dönülmesi gibi karmaşıklık yaratabilecek bir durumun bu yöntem ile önüne geçilmektedir.

Kullanmış olduğumuz YOLOv7 modeli ile yapmış olduğumuz bir denemeye ait çıktılar şu şekilde gözlemlenebilir:





Görsellerdeki turuncu çerçeveler modelin bulmuş olduğu bounding box'ları, eşyayı image içerisinde çerçeveleyebilen en dar dikdörtgeni, ilk ifade bizim asıl ulaşmak istediğimiz şey olan eşyaya atanan etiketi, ikinci ifade ise 0 ile 1 arasında değer alabilen ve birden çok aday etiket olması durumunda kullandığımız “confidence” değerini göstermektedir. Bu deneyde 0.86, 0.87 ve 0.9 gibi confidence değerleri ile bu instance'ların tanınmış olması, bu imageları daha önce hiç görmemiş olan modelin bu yükseklikte bir özgüvenle doğru etiketi atayabilmesi manasına gelip olumlu bir sonuç teşkil etmektedir.

Modelimizin çalışmasını, real-time olarak, uygulamamız içerisinde gözlemlerek için section [4.3.1](#)’in de sonunda ilintilenmiş olan videomuz izlenebilir: (Bahsedilen video altta tekrar ilintilenmiştir.)



(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

4.3.4. Eşyaların Türlerine Ait Etiketlerinin Belirlenmesi Sisteminin Diğer Sistemlerle Entegrasyonu ile İlgili Not

Yukarıdaki section'larda detaylıca açıklanmış olan “Eşyaların Türlerine Ait Etiketlerin Belirlenmesi” sistemi projeye dahil edilmiştir ve başarılı bir şekilde çalışmaktadır. Sistem sonucunda elde edilen etiket, Unity uzayındaki, kullanıcının etkileşime geçtiği GameObject'e, real-time, istenildiği zaman erişilebilecek bir biçimde başarıyla yazılmaktadır. Ancak sistemin, bu etiketin oluşturulma gayesi olan, etiket kullanılarak eşya modellemeleri içerisinde arama yapma ve sonrasında geçici modellemeyi eşya modellemesiyle değiştirme sistemi ile entegrasyonu bulunmamaktadır. Bu sebeple, yukarıdaki sectionlar içerisindeki görsellerde ve videolarda, aynı zamanda da projemizin son halinde bu sistemin çalışması; nesneye etiketin atanması ve kullanıcıya bir UI aracılığıyla etiket atamasının başarıyla gerçekleştirildiğinin gösterilmesi ile sonlanmaktadır. Dolayısıyla, çıktılar da bu şekilde gözlemlenebilmektedir.

4.4. Oyun İçi Model Menüsü & Dinamik Model Yükleme

Kullanıcının istediği 3D modelleri seçebilmesi ve sahnede yaratabilmesi için ilk olarak 3D modellerin bulunduğu bir veritabanı oluşturma fikri ortaya koyduk. Fakat modellerin sayısının ve türünün çok fazla olmasından ötürü, ayrıca ara sunumda da aldığımız geri bildirimle birlikte bu fikrin ölçeklenebilirlik noktasında yetersiz bulunduğu gördükten sonra 3D modellerin yürütme zamanında sorgulanıp, menüde temsili görsellerinin çıkartılıp kullanıcının istediği modeli indirip sahnede yaratabilmesini sağlayan bir yaklaşma yöneldik.

Unity platformunun Assetleri normal şartlarda derleme zamanında beklemesinden ötürü standart işlevleri kullanarak modelleri yürütme zamanında indirerek VR gözlüğün veya bilgisayarın dizininden yükleyip render ederek kullanmamızın mümkün olmadığını gördük. Bu sebeple yaptığımız araştırmaların sonucunda [Zoe for SketchFab](#) [5] isimli pluginin [SketchFab 3D](#) [6] model veritabanına yürütme zamanında sorgu atıp istenilen modelleri indirebildiğini öğrendik. Bu bilgiler ışığında oyun içi menüyü aşağıdaki adımlarda açıklandığı gibi gerçekleştirdik:

4.4.1. Yürütme zamanında model yükleme

Modellerin anahtar kelimelerle sorgu yapılabilmesi için SketchFab API dokümantasyonunda belirtildiği üzere ilk önce ilk sprintte edinmiş olduğumuz API tokeni ile authentication gerçekleştirilir. Sonrasında [4.4.2](#)'de belirtildiği şekilde edindiğimiz model ID kullanılarak model indirilir, render edilir ve modeli barındıran bir game object içine enkapsüle edilir. Bu obje sonrasında obje yaratmak için kullandığımız scriptin yaratılacak obje parametresine set edilir. Böylece kullanıcı menüyü kapattığı zaman odada istediği bir yere tıkladığında obje yaratılabilir.

4.4.2. Model menüsü

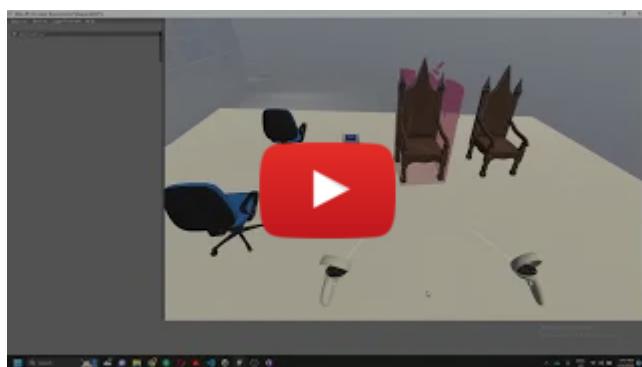
Menüyü Meta'nın Oculus VR için geliştirmiş olduğu assetleri temel alarak geliştirdik. Menüde her eşyaya karşılık gelen kullanıcının controller kullanarak kaydırıp arasından istediğini seçip aktif edebileceği ‘toggle’ isimli bileşenler bulunur. Search keyword

fonksiyonu sesli komut algılandığında ilgili anahtar kelimelerle SketchFab API'a authenticate olduktan sonra modellerden bedava indirilebilir olanları sorgular. Dönen yanıtların her biri için thumbnail temsili görselini ve modeli indirmek için gerekli model UID'yi kaydeder. Sonrasında bu yanılardan her biri için ilgili temsili görseli barındıran toggle elementini yaratıyoruz ve menüye ekliyoruz. Kullanıcı bu görseller arasından istedigini seçebiliyor. 'Download' butonuna bastığında ise model tekrardan bir API isteği ile ilgili modelin indirilip [4.4.1](#)'de anlatılan işlemlerin gerçekleşmesi sağlanır.

Envanter dair bir ekran görüntüsünü aşağıda görebilirsiniz:



Menünün uçtan uca çalıştığı bir videoyu aşağıda seyredebilirsiniz:



(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

4.5. Objelerin Hareket Eşitleme Sistemi

Amacımız menüden seçili odaya konulan eşyaların oda içinde hareket edebilmesini, dönenmesini ve küçülüp büyüyebilmesini sağlamak. Bu sistemi kurabilmek için hazır kütüphaneler pek yardımcı olmadı ve bir nevi kendi kütüphanelerimizi oluşturduk, Oculus'un kütüphaneleri arasına girebilecek kadar iyi bir çıktı elde edildi.

Sistemin özelliklerini madde madde sayacak olursak:

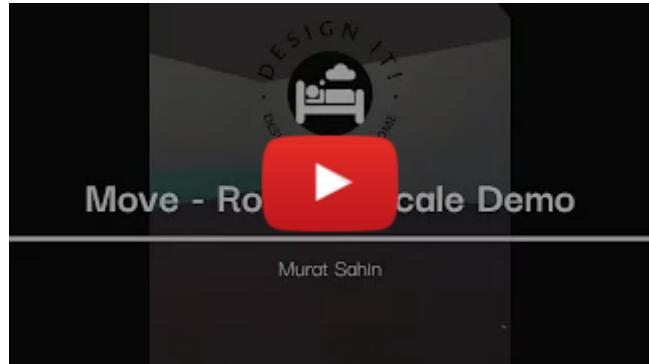
1. Kontrolcü obje üzerine tutulduğunda obje çevresinde kapsayıcı küp çiziliyor.
2. Sağ kontrolcüyle tutulan obje hareket ettirilebiliyor, farklı yüzeylere çarptığında durabiliyor.
3. Sol kontrolcüyle tutulan obje döndürülebiliyor.
4. Hem sağ hem sol kontrolcüyle tutulan obje büyütülüp küçültülebiliyor.

Kullanıcı ister kontrolcüden işin çıkarıp uzaktan, veya objenin yakınına gelip direk temas şeklinde işlevleri gerçekleştirebiliyor.



Objelerin oluşturulma anında gerekli scriptler objeye ekleniyor, dolayısı ile modellerin yürütme zamanında eklenmesi bir sorun değil. Modelin kaynağından bağımsız olarak mümkün olan en az varsayımla geliştirildi, objenin üzerine objenin tüm parçalarını kapsayan en küçük küp oturtuluyor ve onun üzerinden işlem yapılıyor. Ayrıca objenin üzerinde bulunduğu yüzeyin pozisyonu, rotasyonu ve ölçügi gibi değerlerden etkilenmiyor. Bunlardan dolayı önceki başlıklarda da bahsettiğimiz Sketchfab gibi platformlardan alınan modeller için de düzgün çalışıyor.

Aşağıdaki videoda bahsedilen işlevleri gözlemleyebilirsiniz.



(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

Implementasyon Detayları

Oculus Interaction toolkit sayesinde işin çıkarılınca obje üzerinde hangi kontrolcülerin bulunduğu ve hangi kontrolcülerin yakından modifikasyona izin verecek kadar yakın olduğu bilgisini alabiliyoruz. Bu bilgileri kullanarak, her frame obje bilgilerini güncelleyecek şekilde farklı scriptler oluşturduk.

Eşi benzeri çok az veya hiç olmayan bir şekilde yürütme zamanında modelleri oluştururken onlara 10'dan fazla script bağlıyoruz ve böylelikle başarılı bir şekilde her model için bu işlevimizi çalıştırabilmiş olduk.

4.6. Sesli Komut Tanıma & Algılama Sistemi

Benzer uygulamaların ([9,10,11]) hiçbirinde sesli komut özelliğiyle karşılaşmadığımız için, projemize sesli komut özelliği eklemenin, projemizin piyasadaki başarısına önemli bir katkı sunacağını düşünerek hazırladığımız sesli komut uygulamasını üç başlık altında ele aldık. (Tüm komutların algılama ve aksiyona geçme sürecinden önce metine dönüştürülmesi için konuşma tanıma fonksiyonundan geçirilmesi gereklidir.)

4.6.1. Ev Eşyası İstemisiyle Alaklı Komutlar

Bu aşamada kullanıcı istediği ev eşyasını sesli şekilde dile getirdiğinde, fonksiyonumuz komut içerisinde geçen objeyi anlar ve istenilen objenin görsel modeli kullanıcıya sunulur.

4.6.2. Menü ve Oyun Konulu Komutların Algılanması (Projeye entegre edilemedi)

Sistem içerisinde kullanıcının video chat, görüntü, ses, envanter ve menü ile ilgili olan hareketlerini (örn. aç, kapat, menüye dön), sesli komut vasıtasiyla eyleme dökmesi için bu modeli tasarladık. Model yerelde çalışmakta olup sistem entegrasyonunu sağlayamadığımız için kullanmadık.

Sisteme entegre edilebilseydi:

Kullanıcı mikrofonu aktive ettikten sonra istediği komutu sesli şekilde dile getirir.

Komut bu fonksiyon sayesinde algılanır ve kullanıcının istediği eylem otomatik olarak gerçekleştirilir.

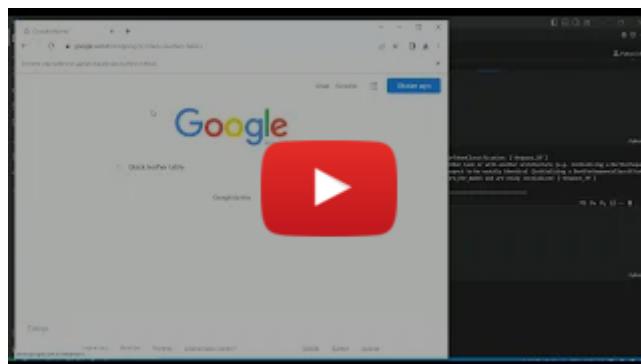
Kullanıcı tarafından fonksiyonun yapamayacağı tarzda komutlar verilirse (örn: Bugün okula git.) sistem hata mesajı döner ve kullanıcından tekrar komut vermesini ister.

4.6.3. Ev Eşyası Piyasa Araması İstemidle Alaklı Komutlar (Projeye entegre edilemedi)

Sistem içerisinde kullanıcının sesli komut vasıtasyyla istediği ev eşyası ürünlerinin piyasadaki örneklerini ve fiyatlarını görebilmesi için bu fonksiyon tasarlandı, fakat yerelde çalışan bu modelin sistem entegrasyonunu sağlayamadığımız için kullanmadık.

Sisteme entegre edilebilseydi:

[4.6.2](#)'de algılanan piyasa bilgisi komutu Google Shopping'de istenilen ürün için gösterilen ilk dört ürünün resmini ve fiyat bilgisini kullanıcıya döner.

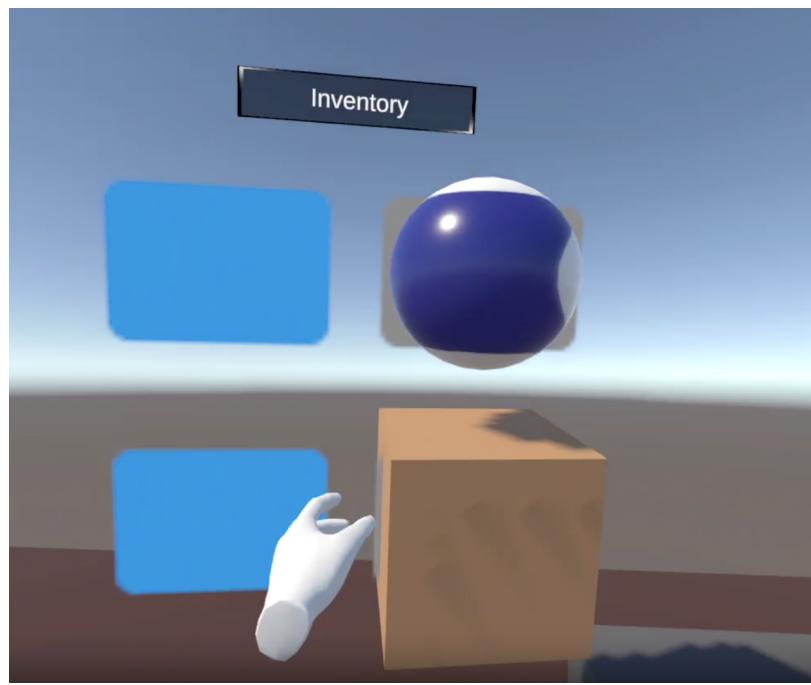


(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

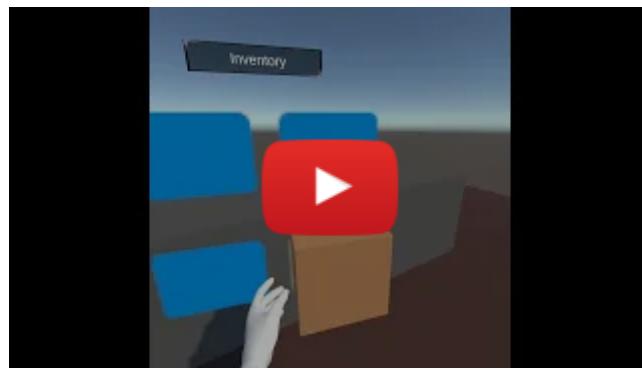
4.7. Inventory & Interact

4.7.1. Inventory

Bu geliştirme kapsamında Inventory adlı, modellemeleri tutabilen dinamik bir liste oluşturuldu. Bu liste sol controller'daki B tuşuna basılarak açılıp kapanabiliyor ve her zaman kullanıcının sol elinin hafif sol-üst bölgesinde; konumu kullanıcıya relativ olarak statik, Unity uzayına göre dinamik bir biçimde güncelliyor.



Kullanıcı buradaki inventory'sinden modellemeleri çekip odaya istediği bir yere koyabiliyor ve/veya isterse odadan bir modellemeyi çekip inventory'sini koyabiliyor. Bu özellik, birden fazla eşyayı aynı anda bir yerden başka bir yere taşımak ve eşyaları yanında tutmak için inventory'nin güzel bir feature'ı. Inventory ile ilgili çekilmiş, kullanımının ve yukarıda bahsedilen özelliğinin gözlemlenebildiği bir video şu şekilde:



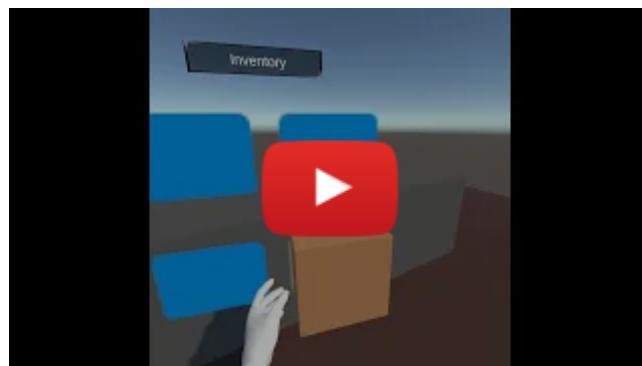
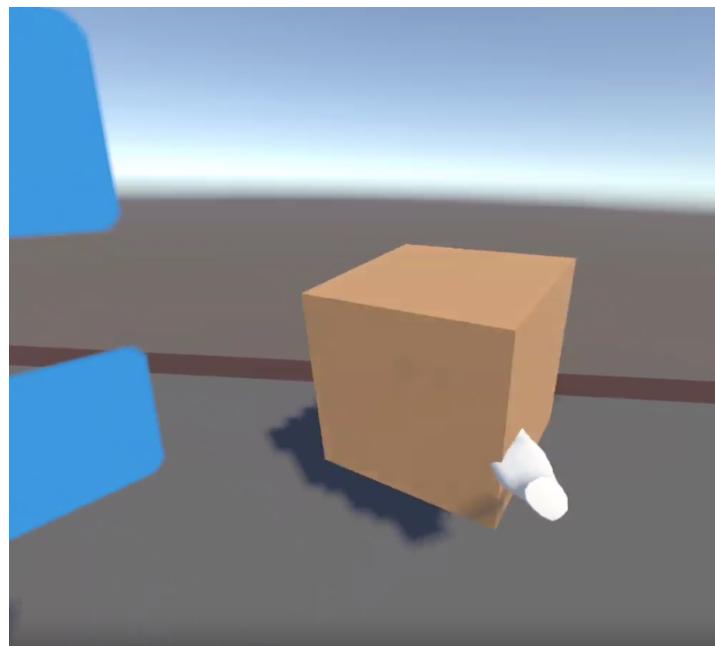
(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

Ancak Inventory'yi geliştirmemizdeki ilk sebebimiz, bahsedilmiş olan birden çok eşyayı tutabilme, taşıyabilme feature'ı değil, odada tespit edilmiş halihazırda var olan eşyaların modellemelerinin uygulama başlangıcında burada görüntülenebilmesi şeklindeydi. Fakat, proje gelişim sürecimizde, önceki ana sectionlardan olan [4.3.](#)'te detaylıca açıklandığı üzere bu iş için bu şekilde bir yapıya ihtiyacımız kalmadı. Dolayısıyla, Inventory sadece birçok eşyayı aynı anda bir yerden başka bir yere taşıma görevi için anlamlı bir eleman haline gelmiş oldu.

Ek olarak, Inventory yapımız, bir sonraki subsection olan [4.7.2.](#)'de anlatılacak Interact kısmında geliştirilmiş olan yöntem ile beraber çalışan bir yapı. [4.7.2.](#)'deki Interact sistemimizin projenin başka kısımlarında geliştirilmiş olan ve yine eşyalarla etkilişim amacı ile çalışan başka sistemlerle beraber ve entegre bir biçimde çalışmıyor olması sebebi ile, Interact ve Inventory sistemleri ile ilgili geliştirmeler halen proje repository'mizde mevcut olsa da, bu noktada bir tasarım kararı olarak geliştirmiş olduğumuz Interact ve Inventory sistemlerimizi kullanmaya devam etmemek kararını projenin sonlarına doğru aldık.

4.7.2. Interact

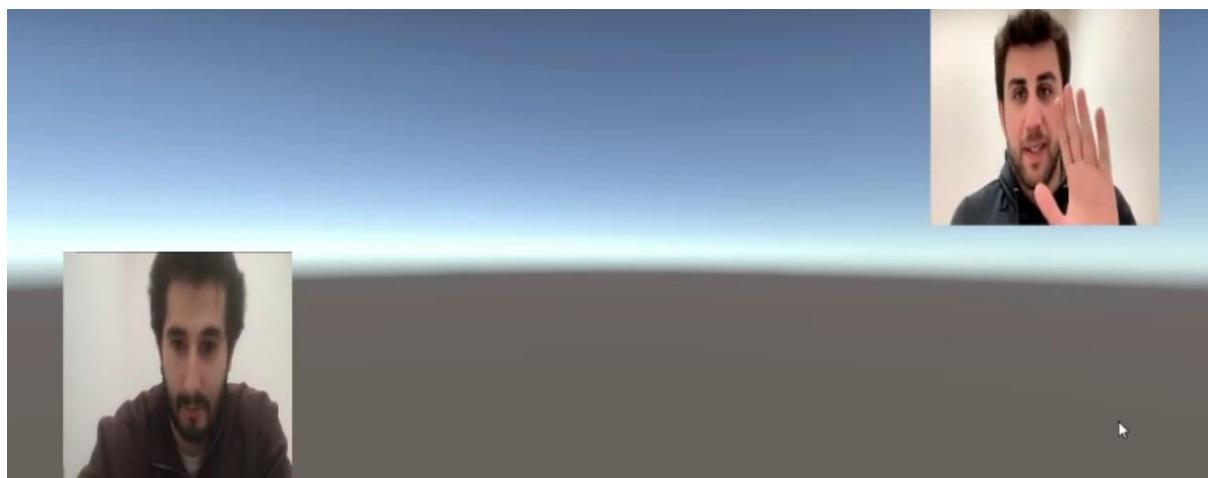
Interact ise, uygulama esnasında odadaki ve Inventory'deki eşyalarla kullanıcının etkileşime geçmesini mümkün kıyan bir sistem. Odadaki eşyaların kullanıcı tarafından ele alınarak bir yerden bir yere taşınmasını, rotate edilebilmesini, yeniden konumlandırılabilmesini mümkün kılmakta. Bunlara basic operasyonlara ek olarak; odadaki eşyaların Inventory'ye konması, Inventory'deki eşyaların Inventory'den alınıp odanın istenilen bir yerine istenilen bir açıyla yerlestirebilmesi gibi işlemleri de sağlayan bir sistem. Bu gibi Interact sistemi ile kullanılabilen uygulama içi bir görüntü ve çekmiş olduğumuz videomuz, [4.7.1.](#) section'ında da paylaşılmış olan video, şu şekilde:



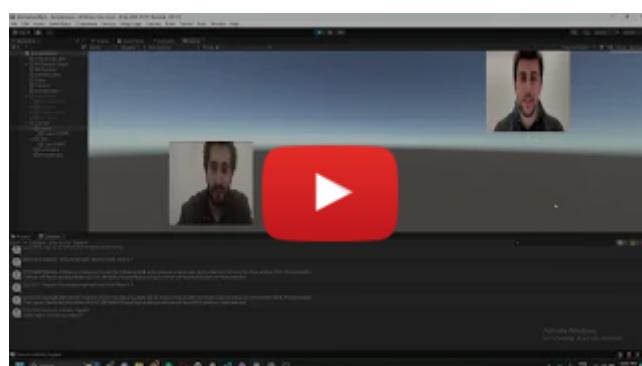
(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

4.8. Video & Voice Chat

Kullanıcı senaryolarından bir başkası olan kullanıcıların odalarını tasarlarken birbirleriyle video chat üzerinden iletişim geçerek fikir alışverişi içinde bulunabilmeleriydi. Bu amaçla [Agora RTC SDK](#) [7] kullanarak iki kullanıcının birbiriyle görüntülü ve sesli olarak konuşabildiği sistemimizi geliştirdik. Bu SDK'nın kullandığımız diğer bileşenlerle versiyon tartışmaları yüzünden maalesef son projeye entegre edemedik ve proof of concept düzeyinde kaldı. Aşağıdaki figürde takım üyeleri Barış Utku Mutan ve Mehmet Can Şakiroğlu'nun konuştuğu bir örnek görebilirsiniz:



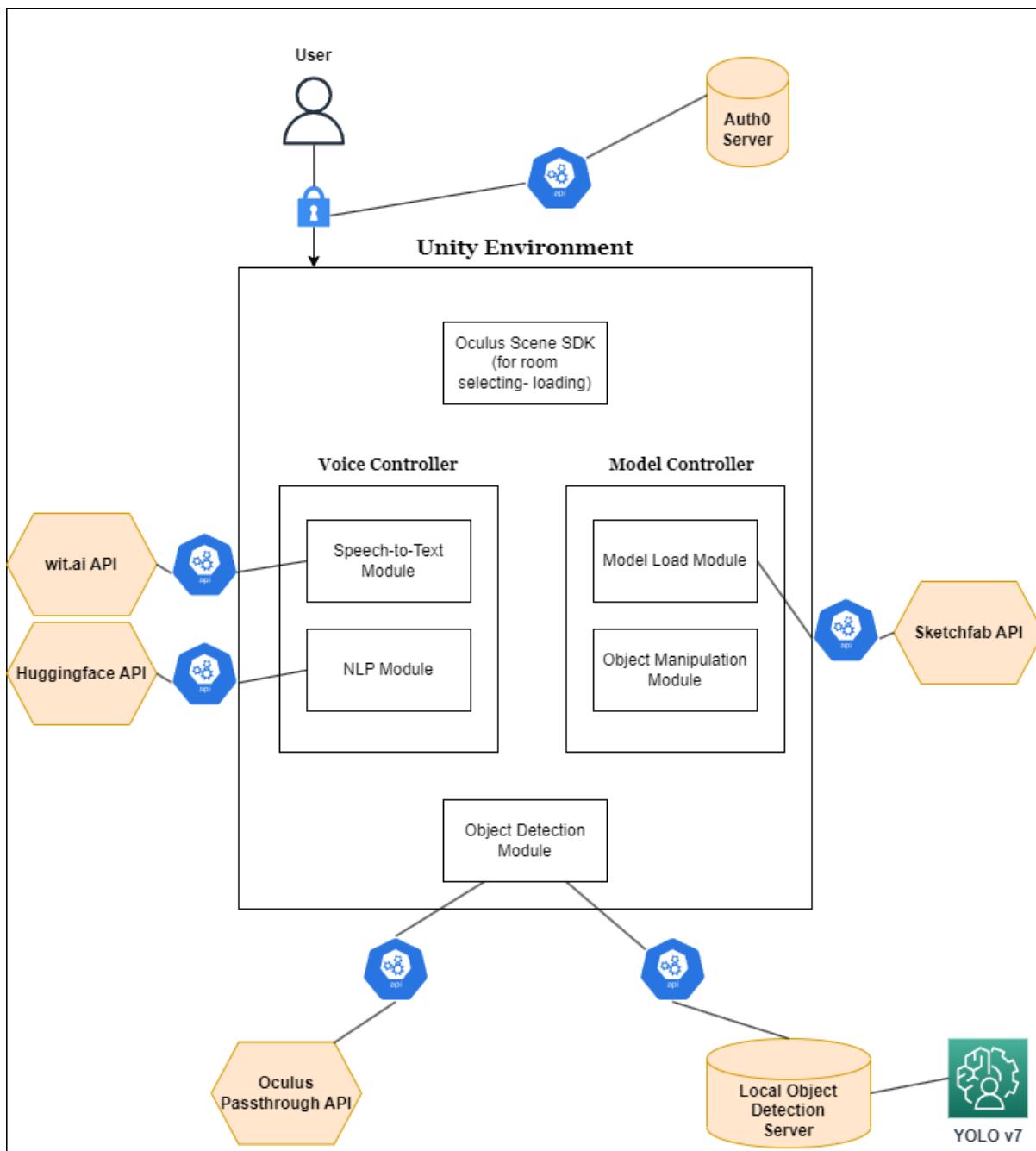
Video konferansın yer aldığı demo videosunu aşağıda izleyebilirsiniz:



(Videoyu açmak için görselin üzerine tıklayıp linke basabilirsiniz.)

5. Sistem Mimarisi ve Sistem Unsurları Arası İlişkiler

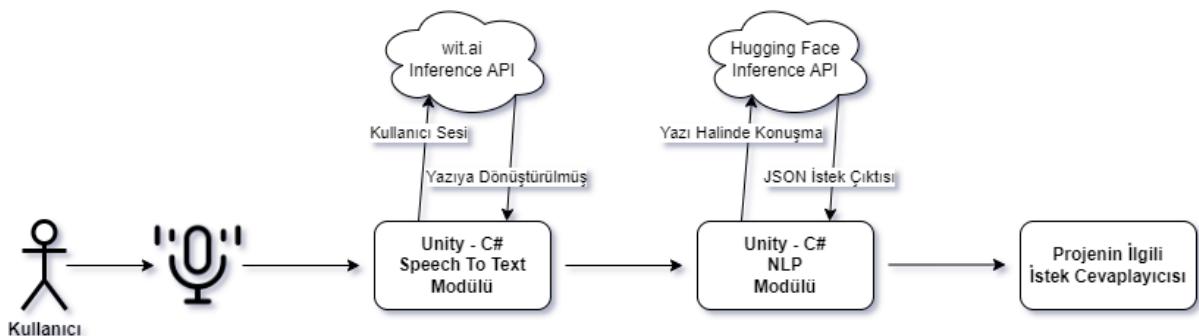
Aşağıdaki şemada genel olarak projemizin çalışma yapısını gözlemleyebilirsiniz.



6. Alt Sistemlerin Detayları

6.1. Doğal Dil İşleme - Sesli Komut

Kullanıcının sesli komutu ile eylemin gerçekleştirilmesine dayalı bu sistem iki ana aşamada ele alınmıştır. İlk aşama ses verisinin metne dönüştürülmesi sürecinden oluşurken ikinci aşamada ise metni anlamlandırma sürecinden oluşur. Genel olarak kurulmuş olan yapıyı aşağıdaki şemada gözlemleyebilirsiniz.



6.1.1. Speech Recognition (Ses verisini metne dönüştürme)

Bu task için başta planımız Kaldi ASR kullanmaktı, fakat daha sonradan Oculus ve Unity işbirliğiyle ortaya çıkan [Voice SDK](#)'i [8] fark ettim. Bu SDK kolayca wit.ai hizmetlerinden yararlanabilmemizi sağlıyor. Sunulan hizmetler arasında da bizim de istediğimiz konuşmadan yaziya dönüştürme işlevi mevcut. SDK tarafından sunulan scriptler ve kendi yazdığımız scriptler yardımıyla beklenen çıktıyi alabiliyoruz.



Böyleslikle scriptlerimiz içerisinde kullanıcı bir butona bastığında ses kaydını başlatıyoruz ve kullanıcı konuşmayı bitirdiğinde otomatik olarak kaydedilen ses verisi wit.ai API'na iletiliyor. Daha sonra gönderilen ses verisinin transkripti uygulamamıza ulaşıyor ve bu veriyi farklı amaçlar için kullanabiliyoruz.

6.1.2. Command Detection (Metinden Aksiyona Geçme)

Deep Learning - Bert Modeli / Furniture object detection

Bu aşamada ilk odaklandığımız durum cumlenin içerisinde geçen ev eşyasını doğru şekilde elde edebilmekti. Bunun için sözlük kullanımı veya sıfat tamlaması tespitinin işe yararlılığı istenilen düzeyde olmadığından transfer learning yaparak Bert- NER (adlandırılmış varlık tespiti) modeli eğittik.

Model için gerekli olan veriyi [ChatGPT](#) [12] kullanarak oluşturduk. Öncelikle ondan içerisinde sıfatlı/sıfatsız ev eşyası içeren 50 cümle istedik, sonrasında ise çıktı cümlelerdeki ev eşyaları yerine [MASK], eşyanın renk sıfatları yerine [COLOR], eşyanın şekil sıfatları yerine [SHAPE] ve son olarak eşyanın materyal sıfatları yerine [MATERIAL] yazmasını istedik. İngilizcede sıfatların çeşidine göre belirli sıralamada kullanılması gereğinden ChatGPT tüm hepsinde kullanımlarına göre aynı sırada etiketleri verdi. Sesli komutta bu sıralamaların yerleri farklı olabileceğiinden (konuşma anında kurallara fazla dikkat edilmediği için) color, shape ve material etiketleri farklı sırada olacak şekilde veriyi kopyaladık. Toplamda 258 veri bu şekilde oluşturulduktan sonra her sıfat ve mask için farklı kelime listeleri oluşturarak bu cümledeki yerlere koyduk ve toplamda 3330 etiketli cümle elde ettik. (Data augmentation- Etiketlerimiz I-O-B tagging formatındadır) Bert (distilbert-base-uncased) modelini oluşturduğumuz veriyle besleyerek elde ettik.

Modeli HuggingFace’e deploy ettiğimizden herhangi bir yerden modeli çağrıabiliyoruz, bu sayede speech recognition kısmında model kullanılabilmıştır.

Rule Based Komut Algılama (kullanılmadı)

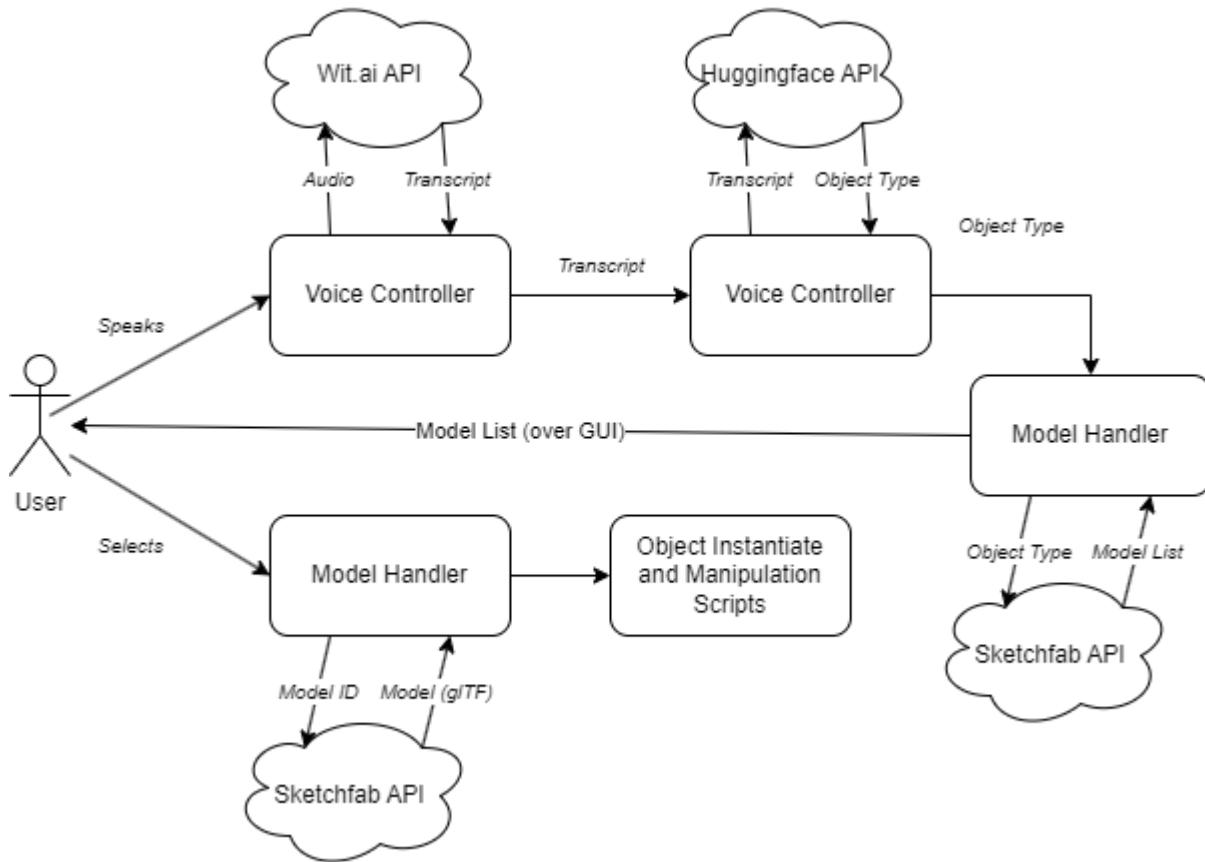
Bu aşamada olası istekleri olası aksiyonları da göz önüne alarak kategorize ettik ve her kategori için içerebileceği kelimelerin listelerini çıkardık. Cümlede belirli kategorilerden kelime bulunuyorsa aksiyon sinyalini veriyor.

Web Search (kullanılmadı)

Bu aşamada kullanıcıya istediği nesnelerin piyasa fiyatlarını dönmek için google alışverişe girmek, görsel ve fiyatlarını çekmek için [Selenium](#) [13] API kullandık.

6.2. Ses ile tetiklenen yürütme sırasında yükleme

[4.4](#) kapsamında bahsedilen, verilen sesli komut ile model listesini çıkarıp seçim yapmayı sağlayan sistemin akış şemasını aşağıda görebilirsiniz.



7. Takım İçi İş Bölümü

Murat Şahin - 191101010

- [4.1](#)'de belirtilen kimlik doğrulama sistemi kurulmuştur.
- [4.2](#)'de belirtilen kullanıcı odasının taranması ve erişilmesi işlevi oluşturulmuştur.
- [4.3](#)'te Local Server-Unity bağlantı tarafında katkı sağlanmıştır.
- [4.4.1](#) yürütme zamanında model yükleme konusunda ön araştırma yapılmıştır.
- [4.5](#) obje manipülasyon sistemi oluşturulmuştur. ([4.7](#)'deki çalışmaların da katkılarıyla)
- [6.1.1](#)'de açıklanan sesten metne dönüştürme işlevi implement edilmiştir.
- [6.2](#)'de açıklanan boru hattı inşa edilmiştir.
- Proje bileşenlerinin birleştirilmesi sağlanmıştır.

Mehmet Can Şakiroğlu - 191101001

- [4.3](#). kapsamında eşya türü etiketi belirlenmesi ve [4.3](#).’ün alt başlıklarında ([4.3.1](#), [4.3.2](#), [4.3.3](#)) deñinilmiş olan, bu konudaki iki farklı yöntemimiz, Object Detection yardımcı otomatik etiket belirleme ve Sesli Komut tabanlı manuel etiket belirleme yapılmıştır.

- [4.7](#) kapsamında ve [4.7](#)'nin alt başlıklarında ([4.7.1](#), [4.7.2](#)) dephinilmiş olan, Inventory özelliği ve Inventory ile uyumlu çalışan Interact özelliği yapılmıştır.
- [4.8](#) kapsamında, test etme ve deneme aşamalarında katkı sağlanmıştır.

Gökçe Başak Demirok - 191401005

[4.6](#)'nın komut anlamlandırma kısmı (4.6.1, 4.6.2 ve 4.6.3 alt başlıkları altında), komut içinde bahsedilen ev eşyasını anlama, uygulama özellikleriyle alakalı komutların sınıflandırılması ve web araması yapılmıştır.

Barış Utku Mutan - 201104082

- [4.4](#) ve alt başlıklarında belirtilen SketchFab API sorgulama, sonuçları oyun içi menüye aktarma ve istenilen modellerin yürütme zamanında indirilip render edilerek odada yaratılabilmesi işlevlerini geliştirmiştir.
- [4.8](#)'de belirtilen Video/voice chat işlevlerini geliştirmiştir.
- Yürütme zamanı yükleme yaklaşımından önce anahtar kelime listesi ile otomatik veritabanı oluşturma işlemini gerçekleştirmiştir.

8. Referanslar

- [1] <https://auth0.com/>
- [2] <https://developer.oculus.com/presence-platform/>
- [3] <https://developer.oculus.com/documentation/unity/unity-scene-overview/>
- [4] <https://github.com/WongKinYiu/yolov7>
- [5] <https://github.com/Zoe-Immersive/SketchfabCSharp>
- [6] <https://sketchfab.com/feed>
- [7] <https://docs.agora.io/en/sdks>
- [8] <https://developer.oculus.com/documentation/unity/voice-sdk-overview/>
- [9] <https://www.arkio.is/learn/>
- [10] <https://rotstudio.com/services/>
- [11] <https://www.inglobetechnologies.com/augmented-reality-architecture/>
- [12] <https://openai.com/blog/chatgpt>
- [13] <https://www.selenium.dev/>