İSTANBUL TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM FAKÜLTESİ

HAREKET YAKALAMA VE SANAL GERÇEKLİK İLE ÇOK OYUNCULU BİR DÖVÜŞ SİMULASYONUNUN GERÇEKLENMESİ

Bitirme Projesi Son Raporu

Umut Yazgan
150130031

Bölüm: Bilgisayar Mühendisliği

Danışman : Asst. Prof. Dr. Gökhan İnce

Ocak 2019

İSTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND INFORMATICS

IMPLEMENTATION OF A MULTIPLAYER COMBAT SIMULATOR USING MOTION CAPTURE AND VIRTUAL REALITY

Graduation Project Final Report

Umut Yazgan
150130031

Department: Computer Engineering
Division: Computer Engineering

Advisor : Asst. Prof. Dr. Gökhan İnce

January 2019

Özgünlük Beyanı

Burada ilan ediyorum ki bu çalışmada,

1. Bütün dışarıdan alınmış referanslar açıkça ve detaylıca kaynakları belirtilerak alıntılanmış,
2. Geriye kalan bütün bölümler, özellikle bu çalışmanın temelini oluşturan teorik çalışmalar ve yapılan yazılımsal/donanımsal katkılar şahsımca yapılmıştır.

İstanbul, 4 Ocak 2019

Umut Yazgan

HAREKET YAKALAMA VE SANAL GERÇEKLİK İLE ÇOK OYUNCULU BİR DÖVÜŞ SİMULASYONUNUN GERÇEKLENMESİ

(Özet)

Savaş sanatları, Dünya üzerinde çok sayıda insan tarafından gerek kendini savunma amaçlı gerek spor olarak gerekse profesyonel bir biçimde çalışılmaktadır. Kimi savaş sanatlarının, özellikle de kılıç, bıçak gibi kesici aletlerle çalışılanların, tam hız ve temas ile güvenli biçimde çalışılması mümkün değildir. Bu noktada, kesici aletlerde daha yavaş ve temassız çalışmaya alternatif bir yol olarak, bir bilgisayar simulasyonundan fadyalanılabilir.

Bu projede, dövüş simulasyonunun kapsamı biraz daha dar tutularak, iki kişilik bir kılıç dövüşü simulasyonu gerçeklenmiştir. Simulasyonda, iki dövüşçü hareket yakalama kıyafetleri ve sanal gerçeklik gözükleri kullanarak, ağ üzerinden, birbirlerinin yanında olma ihtiyacı olmaksızın karşı karşıya gelip çalışma imkanı bulabilirler. Bunun sağlayacağı bir diğer avantaj ise, çalışmak için aynı yerde olma zorunluluğunu ortadan kaldırmış olması ve savaş sanatı çalışan insanlara dojo çalışmalarının haricinde evlerinde de karşılıklı çalışma imkanı sunmasıdır.

Sistemin gerçeklenmesi için Perception Neuron hareket yakalama cihazı, Cardboard sanal gerçeklik gözlüğü ve Android telefonlarla çalışacak bir Unity uygulaması yazılmıştır. Kullanıcı Perception Neuron hareket yakalama cihazını giyer ve bir bilgisayarda çalışan Axis Neuron yazılımına bağlar. Axis Neuron, hareket verisini BVH formatında ağ üzerinden yayınlar, böylece iki kullanıcı da kendi Android cihazlarından hem kendi Axis Neuron'larının BVH yayınına hem de diğer kullanıcınınkine bağlanır ve hem kendi hareket verilerini hem de diğer kullanıcının hareket verisini simulasyon içerisindeki bir insan modeline(bir avatara) aktarırlar.

Söz konusu insan modellerinin ellerinde birer adet basit kılıç benzeri silah vardır. Bu silahların keskin kısımları, kullanıcı avatarlarından herhangi birine temas ettiği takdirde, o avatar kesilmiş ve ölmüş sayılır. Axis Neuron'la bağlantısı ve simulasyon görüntüsü kısa süreli kesilir, sonrasına simulasyona geri döner ve çalışma devam eder.

Kullanıcılar tüm bu simulasyonu Android cihazlarının ekranından bir sanal gerçeklik gözlüğü ile gözlemlerler.

IMPLEMENTATION OF A MULTIPLAYER COMBAT SIMULATOR USING MOTION CAPTURE AND VIRTUAL REALITY

(Summary)

Marial arts are being practiced by many people around the world for various reasons including self-defence, sports and as a profession. Some martial arts, especially ones that utilize sharp weapons such as knives and swords, cannot be safely practiced with full speed and contact. At this point, a computer simulation can be used as an alternative for slow and no-contanct practice.

In this project, scope of the combat simulation is kept small and a two person sword fighting simulation has been implemented. In simulation, two fighters can come together and practice online without the need to be near each other, with help of motion capture gear and virtual reality googles. Another advantage provided by this is that it will remove the necessity to be at the same location to study and give martial artists an opportunity to practice outside of dojo, at their homes.

For implementation of the system, a Unity application what works with Perception Neuron motion capture device, Cardboard virtual reality googles and Android phones. The user wears Perception Neuron motion capture device and connects it to Axis Neuron software that runs on a PC. Axis Neuron broadcasts motion data to network in BVH format so that two users can subscribe to both their and other users Axis Neuron's BVH broadcast, allowing them to send both users' motion data to a humanoid model (an avatar) in simulation.

These said humanoid models each have a simple sword-like weapon in their hands. If sharp parts of these weapons contacts any of the user avatars, that avatar is considered sliced and dead. It's connection with Axis Neuron and it's vision is interrupted for a short period, then it returns to the simulation and keeps training.

Users observe this whole simulation from display of their Android devices, using a virtual reality headset.

İçindekiler

İçindekiler

1 Giriş ve Proje Özeti....5
2 Literatür Taraması....7
3 Hedeflenen Tasarım ve Gerçeklenen Sistem....8
3.1 Kullanılan Teknolojiler ve Entegrasyonları....8
3.1.1 Terminoloji....8
3.1.2 Perception Neuron....8
3.1.3 Cardboard....10
3.1.4 Unity....11
3.1.5 Entegrasyon....12
3.2 Proje Yazılımı....13
3.2.1 Lobi....13
3.2.2 Simulasyon Çevresi....14
3.2.3 Çarpışmalar ve Fizik Motoru....16
3.2.4 Ağ Üzerinden Kullanıcıların Bağlanması....17
3.2.5 Simulasyon Akışı ve Mekanikler....19
3.2.6 Sanal Gerçeklik....21
4 Test Ortamı ve Testler....23
4.1 Test Ortamı....23
4.2 Testler....23
5 Kıyaslamalı Değerlendirme ve Tartışma....24
6 Sonuç ve Gelecek Çalışmalar....25
7 References....26

1 Giriş ve Proje Özeti

Giyilebilir teknoloji, hareket yakalama ve sanal gerçeklik çalışmaları son yıllarda ciddi gelişmelere sahne olmuştur. Giyilebilir teknolojiler; akıllı saatlerden bileklik ve gözlüklere farklı noktalardan hem son kullanıcı tarafından daha kolay satın alınabilir olmasıyla hayatımızda yer etmektedir, hem de donanım ve yazılım geliştiriciler bu teknoloji üzerine çalışmalarını arttırmaya devam etmektedirler. Hareket yakalama teknolojileri ise, bir yandan Xbox Kinect, Nintendo Wii veya Switch gibi son kullanıcı tarafından edinilebilir oyun teknolojileri ile hayatımıza girerken, bir yandan da akademide araştırmacılar için bir ilgi odağı haline gelmiştir. Kinect kamerasının robotik araştırmalarında kilit öneme sahip bir donanım hale gelmesi bunun için güzel bir örnektir. Bunun ötesinde bu projede kullanılan Perception Neuron gibi daha gelişmiş giyilebilir hareket yakalama cihazları ise henüz son kullanıcılar arasında popüler olacak kadar satın alınabilir olmamalarına karşın araştırmacı ve üreticiler arasında yaygın kullanımları mevcuttur. Sanal gerçeklik teknolojileri ise, Android cihazlarda kullanılabilecek Google Cardboard gibi basit ve ucuz çözümler sayesinde hatırı sayılır bir popülerliğe ulaşmıştır. Bütün bu teknolojiler eğlenceye ve tüketime yönelik alanların ötesinde kullanımlara sahiptir. Örneğin hareket yakalama teknolojileri film üretiminde ve hasta takibinde[1], sanal gerçeklik ise psikolojik tedavi süreçlerinde[2] kullanılabilecek teknolojilerdir.

Bu projede ise, hareket yakalama ve sanal gerçeklik teknolojilerinin bir arada savaş sanatları eğitiminde kullanılması amaçlanmaktadır. Önerilen sistemde; Perception Neuron hareket yakalama kıyafetlerini ve bir sanal gerçeklik gözlüğü giyen kullanıcılar, kendi bulundukları konumdan Android telefonları için geliştirilmiş uygulamayı kullanarak internet üzerinden bağlantı kurabilecek ve Perception Neuron tarafından algılanan hareket verilerini internet üzerinden birbirlerinin telefonlarına aktararak karşılıklı antrenman yapabileceklerdir.

Böyle bir uygulamanın savaş sanatları eğitim ve çalışmlarına birden fazla katkısı olacaktır. Öncelikli olarak, çalışmak isteyen kişilerin fiziksel olarak aynı noktada bulunma ihtiyacı ortadan kalkacaktır. Bu sayede bir dojoda yapılacak çalışmalara ek olarak, şahıslar evlerinde geçirdikleri zaman diliminde de bu uygulamayı kullanarak karşılıklı çalışma imkanına sahip olacaktır. Bir araya gelme imkanı olmayan, başka şehir veya ülkelerde yaşayan insanlarla da karşılıklı olarak çalışmak mümkün hale gelecektir. Çalışma sürecinin bir oyun formatına sokulması ise, yeni başlayan sporcular, özellikle de çocukar için bir motivasyon kaynağı haline gelecektir. Bunun yanı sıra, bazı savaş sanatlarındaki kimi tehlikeli tekniklerin çalışılması bu şekilde kolaylaşacaktır. Örneğin, aikidoda kılıç veya bıçaklarla çalışılan teknikler dojoda yüksek hızda ve tam temas ile çalışılamamaktadır. Bu tekniklerin çalışılması için bu uygulama alternatif bir yol sunacaktır. Tüm bunlara ek olarak belirtmek gerekir ki, uygulamanın sonuç olarak amacı dojo çalışmasının yerini almak değildir. Uygulama, savaş sanatları çalışması için olmazsa olmaz olan karşılıklı ve tam temaslı dojo çalışmasına paralel olarak kullanılabilecek, kendince artı ve eksi yönleri olan alternatif bir çalışma metodu olarak geliştirilmiştir.

Uygulamanın gerçeklenmesi için Unity oyun geliştirme platformu kullanılmıştır. Perception Neuron hareket yakalama cihazından anlık olarak alınan hareket verileri, Windows işletim sistemi üzerinde çalışan Axis Neuron isimli yazılım aracılığı ile yayınlanmış ve Perception Neuron için geliştirilmiş Unity SDK'sı kullanılarak Unity üzerinden bu veri alınıp, simulasyon içerisinde kullanıcıya ait bir insan modelinin hareket ettirilmesi için kullanılmıştır. Uygulama, Android platformu için geliştirilmiştir ve

kullanıcı, Android cihazını bir sanal gerçeklik gözlüğüne yerleştirerek, hareketlerini aktardığı insan modelinin görüş açısından çalışma alanını ve internet üzerinden bağlı olduğu çalışma arkadaşının insan modelini görmektedir.

Sonuç olarak elde edilecek projede hedef, iki kullanıcının karşılıklı olarak mümkün olduğunca az hata ve gecikme olan gerçekçi bir simulasyon ortamında antrenman yapabilmesidir. İlk aşamada kılıç kullanımına dayalı bir simulasyonun gerçekleşmesi amaçlanmıştır, ancak ilerleyen geliştirmelerle yumruk, tekme, kollarla blok gibi farklı saldırı ve savunma tekniklerinin kullanılabileceği bir simulasyon gerçeklenmesi hedeflenmektedir.

2 Literatür Taraması

Sanal gerçeklik ve hareket yakalamanın bir arada kullanıldığı pek çok proje mevcuttur. Rincon, Yamasaki ve Shimoda bu teknolojilerin EMG analizi ile beraber rehabilitsasyon süreçlerinde kullanılması üzerinde bir çalışma yayınlamışlardır. Çalışmalarında, YEI 3-Space hareket yakalama cihazı ve Oculust Rift sanal gerçeklik gözlüğü kullanarak ağır travma geçişrmiş hastaların rehabilitasyon süreçlerini kolaylaştıracak bir oyunu Unity'de geliştirmişlerdir. Oyun oynanırken hastanın durumunu takip edebilmek için EMG sinyal analizi kullanılmıştır. [2]

Sanal gerçeklik ve hareket yakalama cihazlarının dans eğitimi için kullanılmasını ise Chan, Leung, Tang ve Komura önermiştir. Geliştirdikleri sistemde, sanal gerçeklik ortamında bir dans eğitmeni simule edilmiştir. Bu sanal eğitmen, öğrencilere çeşitli dans hareketleri gösterir ve öğrenciler bu hareketleri bir hareket yakalama kıyafeti ile taklit ederler. Bu hareket yakalama kıyafetinden gelen veriye bakarak hareketin ne kadar doğru yapıldığını ölçen sanal eğitmen bu sayede öğrenciye anında geri bildirimde bulunabilir. [3]

Royston, DeFanti ve Perlin ise, geliştirdikleri GraphiteVR isimli yöntem ile Samsung GearVR sanal gerçeklik gözlüğü ve Perception Neuron hareket yakalama cihazı kullanarak bir veri görselleme sistemi oluşturmuşlardır. Veri görselleme, normalde 3 boyutlu gösterilmesi gereken verilerin iki boyutlu ekranlara yansıtılması sebebiyle hep bir sorun olurken, bu sorunu veriyi sanal gerçeklik ortamında gösterip, hareket yakalama ile manipule etme imkanı sağlamıştır oluşturdukları sistem. [4]

Kameralı bir hareket yakalama sisteminin hasta takibinde kullanılması ise Ye, Ci, Katsaggelos ve Liu tarafından önerilmiştir. Çalışmalarında, birden çok kameradan faydalanan bir sistemle hareket bir hareket algılama sistemi oluşturularak hastaların hareketliliklerinin otomatik fark edilebilmesi sağlanmıştır. Bu sayede, sağlık çalışanları hastanın başında beklemeden de takibini yapabileceklerdir. [1]

Son olarak bu projenin ilham aldığı bir diğer çalışma ise, Bilgin tarafından geliştirilen Hareket Yakalama ile Sanal Gerçeklikte Tenis oyunu projesidir. Bu projede Bilgin, Perception Neuron hareket yakalama cihazı ve Samsung GearVR sanal gerçeklik gözlüğünden faydalanan bir tek kişilik tenis oyununu Unity oyun motoru kullanarak geliştirmiştir. [5]

3 Hedeflenen Tasarım ve Gerçeklenen Sistem

Bu bölümde, öncelikle projede kullanılan teknolojiler ayrıntılı bir biçimde anlatılacaktır. Ardından bu teknolojilerin nasıl birbiriyle entegre edildiği ve sonuçta oluşan projenin gerçeklenmesinde kullanılan yazılım metotları ayrıntılı olarak açıklanacaktır.

3.1 Kullanılan Teknolojiler ve Entegrasyonları

3.1.1 Terminoloji

Bu bölümde bu raporda sıkça kullanılacak bazı terimlerin tanımlarına yer verilmiştir. Terimlerin çoğunluğu Unity oyun motoru ile ilgilidir.

Asset: Bir Unity projesinde kullanılan her türlü varlığa verilen genel bir isimdir. [6] Bu raporda genellikle Unity Asset Store veya Perception Neuron Unity SDK'sı gibi dış kaynaklardan edinilmiş parçalardan bahsederken kullanılmıştır.
Collider: Unity'de bir nesnenin çarpışmalarda yer alabilen kısmına verilen ad. Nesneye bir component olarak eklenir ve kapsül, küp veya "mesh" gibi çeşitli şekilleri olabilir. [7]
Component: Unity'de GameObject'lerin üzerine eklenen ve eklendikleri nesnelere çeşitli özellikler katan "parça"lar. Betikler, rigidbody'ler, collider'lar birer component örneğidir. [8]
GameObject: Unity'de bir sahne içerisindeki nesnelere verilen genel ad. Kendi başlarına varolmaktan öte pek bir işlevleri olmamakla birlikte, içlerinde barındırdıkları component'lerle birlikte işlevsel bir bütün olurlar. [9]
Mesh: Bir arada belirli bir 3 boyutlu şekil oluşturan bir poligon kümesi. [10] Bu raporda genellikle "mesh collider"lardan bahsederken kullanılmıştır. Herhangi bir 3 boyutlu nesne şeklinde olabilrmesi için poligonlarla inşa edilmiş colliderlara mesh collider denir. Yine raporda adı geçen convex mesh collider'lar ise dış bükey 3 boyutlu şekil oluşturan mesh'ler kullanılarak yapılmış collider'lara denir.
MonoBehaviour: Bütün Unity scriptlerinin kalıtıldığı(inherit edildiği) temel sınıf. [11]
NetworkBehaviour: MonoBehaviour'dan kalıtılmış, MonoBehaviour yerine kullanıldığı zaman betik içerisinde Unity'nin ağ özelliklerinin kullanılmasına olanak sağlayan sınıf.
Prefab: Kompleks GameObject'lerin birer kalıp halinde kaydedilip yeni nesnelere iskelet olarak tekrar tekrar kullanılmasına olanak sağlayan bir tür asset. [12]
Rigidbody: Bağlı olduğu nesnenin Unity fizik motorundan etkilenmesini sağlayan component. [13]
SyncVar: NetworkBehaviour'dan kalıtılmış betiklerde kullanılabilen, sunucu ve istemciler arasında senkronize olan bir değişken türü. [14]

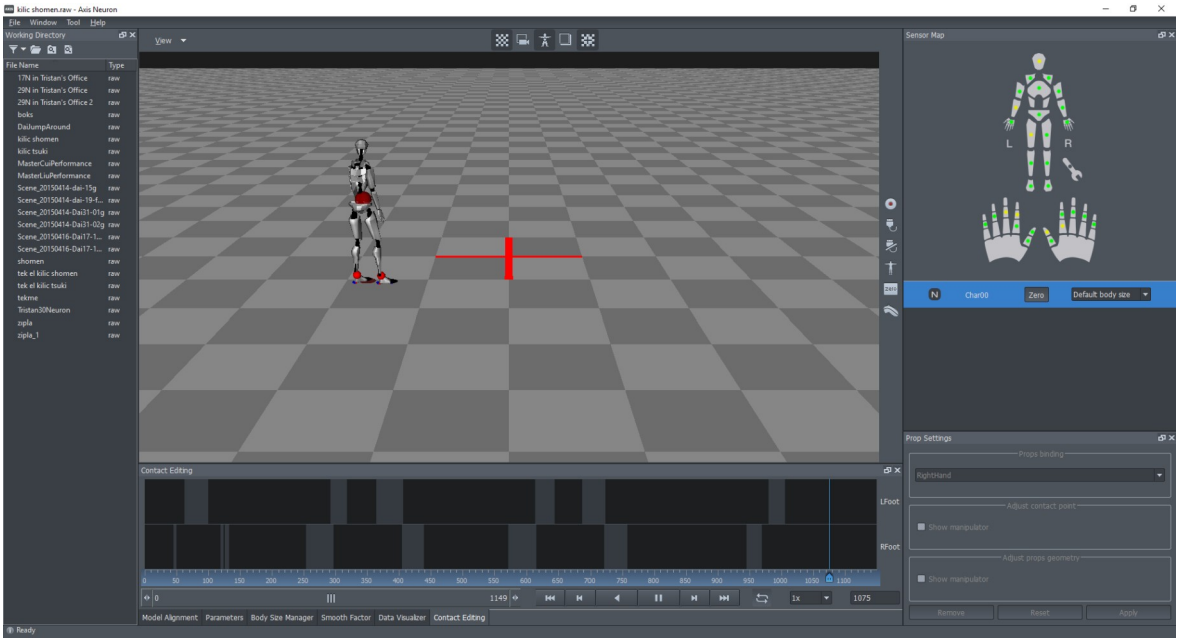
3.1.2 Perception Neuron

Perception Neuron, Noitom tarafından geliştirilmiş bir giyilebilir hareket yakalama cihazıdır. Cihaz, gövde ve uzuvlara yerleştirilen "Neuron" adı verilen sensörler ve bu sensörlerden gelen veriyi toplayıp Wi-Fi veya USB aracılığı ile yayınlayan bir merkezden (hub) oluşmaktadır. [15] Neuronlar, 3 boyutlu ölçüm yapabilen ivmeölçer, jiroskop ve manyometre sensörleri içeren aşağı yukarı yarım santimetre boyutunda cihazlardır. [5]

Perception Neuron, kullanıcının bu Neuron sensörlerinden 32 tanesini Görsel 3.1'de görüldüğü gibi vücuduna yerleştirmesine olanak sağlar.

Görsel 3.1: Perception Neuron hareket yakalama cihazı ve Cardboard sanal gerçeklik gözlüğü

Görsel 3.2'de, Perception Neuron'un, hareket verisinin çeşitli şekillerde işlemek ve yayınlamak için kullandığı Axis Neuron isimli yazılım görülmektedir. Axis Neuron, Perception Neuron'dan USB veya Wi-Fi ile aldığı veriyi yazılım içerisindeki bir insansı figürü oynatmak için kullanır. Bu figürün boyut, eğim vs gibi kimi parametreleri program içerisinde ayarlanabilir. Axis Neuron, hareketleri kaydedip Perception Neuron bağlantısı yokken oynatma imkanı sunar. Aynı zamanda, gelen hareket verisini BVH formatında başka yazılım ve donanımların kullanabilmesi için ağ üzerinden yayınlayabilmektedir.



Görsel 3.2: Axis Neuron yazılımı

Perception Neuron, aynı zamanda bir asset paketi halinde bir Unity SDK'sı sunmaktadır. Bu SDK içerisinde Axis Neuron'da gözüken insansı modelin Unity'de kullanılabilir çeşitli versiyonları ve bu modelleri Axis Neuron'a bağlayıp hareket verisini takip etmelerini sağlayabilmek için gerekli kimi betikler mevcuttur. Bu betikler aracılığı ile, Axis Neuron'un ağ üzerinden yaptığı BVH formatında hareket verisi yayınını Unity SDK'sında bulunan NeuronRobot modellerinden biri veya uygun iskelet yapısına sahip başka herhangi bir insansı model takip edebilir ve hareketleri Unity içerisinde aynen taklit edebilir.

3.1.3 Cardboard

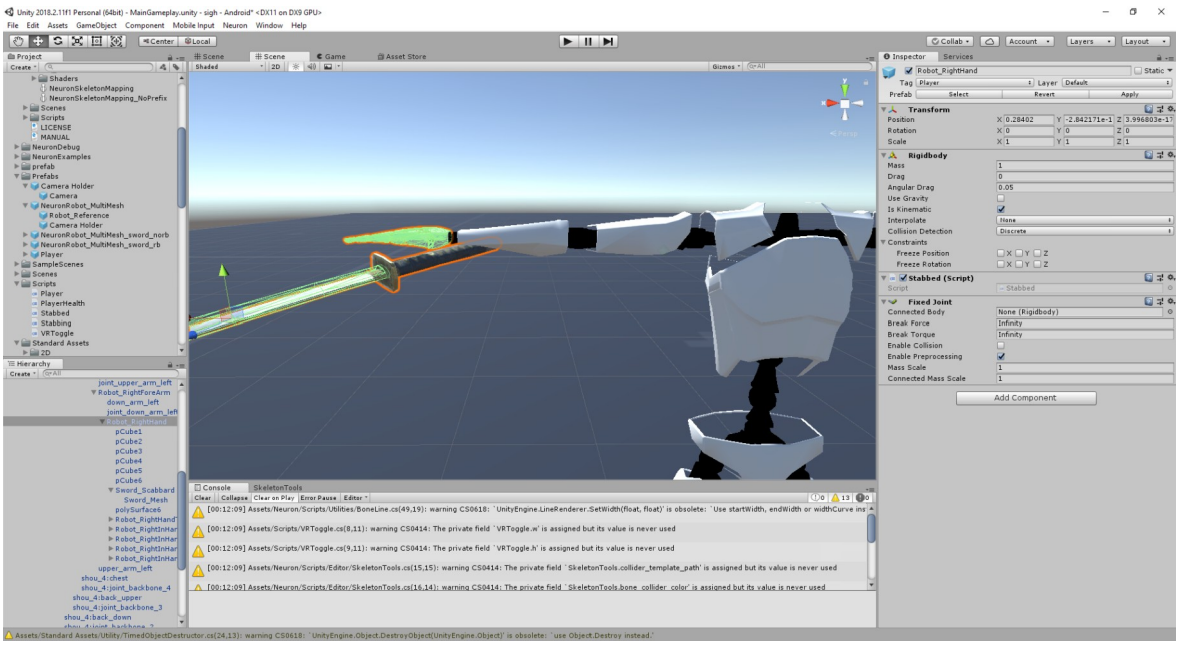
Google Cardboard; kartondan yapılma, son derecec düşük maliyetli, mobil cihazlarla kullanılabilen ve Google'ın geliştirdiği sanal gerçeklik API'si ve SDK'larından faydalanılarak programlanabilen bir sanal gerçeklik gözlüğüdür. Gözlüğün yapımı son derece basit olduğu için sayısız üretici tarafından çeşitli varyasyonları yapılmaktadır, hatta evde bile yapılması mümkündür. Bu projenin geliştirilmesinde de Görsel 3.3'de gördüğünüz son derece basit gözlük kullanılmıştır.

Görsel 3.3: Projede kullanılan Cardboard sanal gerçeklik gözlüğü

Cardboard içine bir mobil cihaz yerleştirilerek kullanılmaktadır. Bu proje geliştirilirken, Sony Xperia Z5 Compact Android akıllı telefon bu amaçla kullanılmıştır. Günümüzde tüm akıllı telefonlarda bulunan ivme ölçer ve jiroskop aracılığı ile uygulama kullanıcının kafa hareketlerini algılamakta ve bakış açısını ona göre ayarlamaktadır.

3.1.4 Unity

Unity, Unity Technologies tarafından geliştirilen ve çok sayıda platforma destek veren bir oyun motorudur. 2 boyutlu ve 3 boyutlu oyun, animasyon veya simulasyon geliştirmek için çok sayıda içerik üretici tarafından aktif biçimde kullanılmaktadır. Sürükle-bırak formatında araçlarla uygulama geliştirmeye olanak sağlayan editör arayüzünün yanı sıra, C# programlama dilinde Unity'nin geliştiricilere sunduğu API'ler yardımıyla kod yazarak uygulama geliştirmek de mümkündür.[16] Görsel 3.4'de Unity'de projenin geliştirilme sürecinden bir ekran alıntısı görülmektedir.

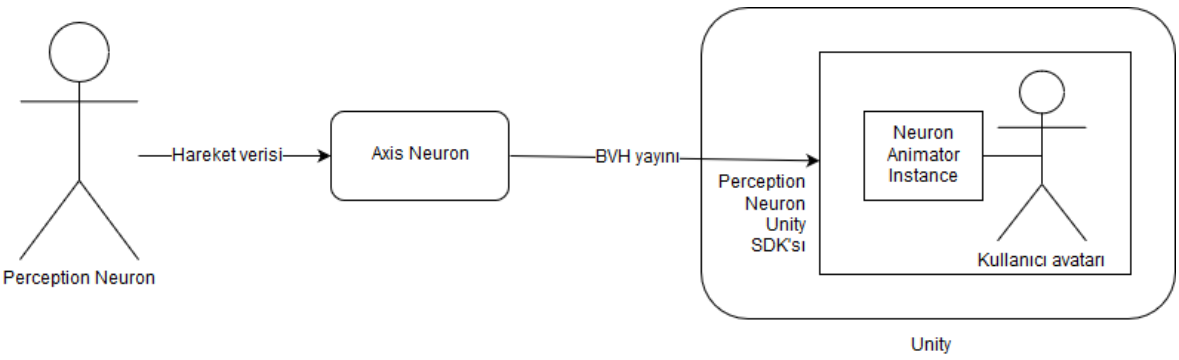


Görsel 3.4: Unity'de projenin geliştirilme sürecinden bir görüntü

Google'ın Cardboard cihazlar için sağladığı SDK ve Unity'nin kendi XR API'si projede sanal gerçeklik ortamının oluşturulması için kullanılmıştır. Projenin Android cihazlara yüklenip çalıştırılabilecek APK formatında derlenmesi için Android Studio ile birlikte gelen Android SDK'sı ve Java Development Kit (JDK) kullanılmıştır. Perception Neuron ile entegrasyon ise, Perception Neuron için geliştirilmiş ve Unity'ye Asset olarak eklenebilen Unity SDK'sı aracılığı ile gerçekleştirilmiştir.

3.1.5 Entegrasyon

Projede kullanılan tüm donanım ve yazılımın entegrasyonu için izlenen süreç, Görsel 3.5'te verilen akış diyagramında gösterilmiştir.



Görsel 3.5: Sistem parçalarının entegrasyonu

Öncelikle, kullanıcının giydiği Perception Neuron hareket yakalama cihazından alınan hareket verisi, kullanıcının bilgisayarında çalışan Axis Neuron yazılımına USB veya Wi-Fi üzerinden aktarılmaktadır. Axis Neuron, topladığı hareket bilgisini ikili(binary) BVH formatında bilgisayarın 7001 portundan yayınlamaktadır. Burada önemli bir ayrıntı, farklı yönlendiricilere (router) bağlı uzak bilgisayarlar arasında simulasyonun çalıştırılabilmesi

için 7001 portunun dışarıdan erişime açılmış (yönlendirilmiş) olması gerekliliğidir. 7001 portundan yayınlanan BVH formatındaki hareket verisi, Android cihazda çalışmakta olan ve Unity ile geliştirilmiş simulasyon tarafından alınır ve kullanıcının simulasyon içerisindeki avatarını hareket ettirmek için kullanılır. İki kullanıcı da birbirlerinin Axis Neuron'larından gelen veriyi eş zamanlı olarak takip ederler. Kullanıcılar, simulasyon içerisinde avatarlarının kafası yerine yerleştirilmiş kameralardan gelen görüntüyü, sanal gerçeklik gözlüklerini kullanarak görebilir ve kafalarını hareket ettirerek ortamda çevrelerine bakabilirler.

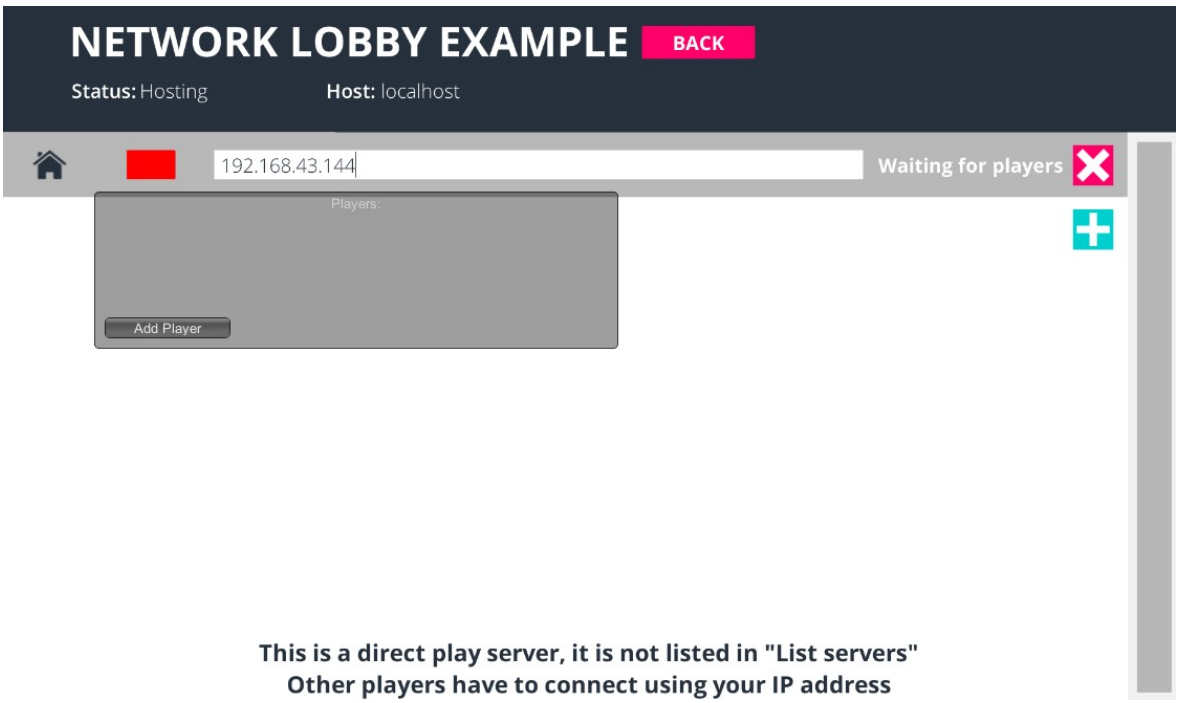
3.2 Proje Yazılımı

Proje yazılımının üretiminde, Unity'nin sağladığı sürükle-bırak tarzı araçlar kullanılarak bir simulasyon ortamı oluşturulmuş, bu ortamda Perception Neuron Unity SDK'sı tarafından sağlanan VR destekli insan modeli modifiye edilerek kullanılmış ve kullanıcıların bu modelleri kontrolü ve birbiri ile etkileşimini sağlamak için C# programlama dilinde betikler yazılmıştır. Bunların yanı sıra, modifiye edilmiş kimi hazır Assetlerden de faydalanılmıştır.

3.2.1 Lobi

Uygulama başlatıldığı zaman, kullanıcıları bir lobi ekranı karşılamaktadır. Bu lobi ekranı, Unity Asset Store'dan ücretsiz edinilebilen ve Unity Technologies tarafından örnek sağlaması amacıyla geliştirilmiş Network Lobby assetinin bir miktar ekleme yapılmış halidir.

Uygulamada, lobinin DIRECT PLAY başlığı altındaki PLAY AND HOST sekmesi kullanılmaktadır. Bu sekme kullanılarak kullanıcılardan biri oyun kurucu (host) rolünü üstlenir. Diğer kullanıcı, oyun kurucu olan kullanıcının, aynı ağ üzerindelerse dahili, farklı ağlara bağlı iseler harici IP adresini JOIN A GAME kısmındaki kutucuğa yazıp JOIN butonuna tıklayarak lobiye dahil olabilir. Görsel 3.5'de görülen yeni ekranda ise kullanıcılar, bağlanacakları Axis Neuron BVH yayınının IP adresini kendilerine ait olan kutucuklara yazıp yanındaki JOIN butonuna tıklarlar. İki kullanıcı da JOIN butonuna tıkladıktan sonra 3 saniye içerisinde simulasyon başlar.



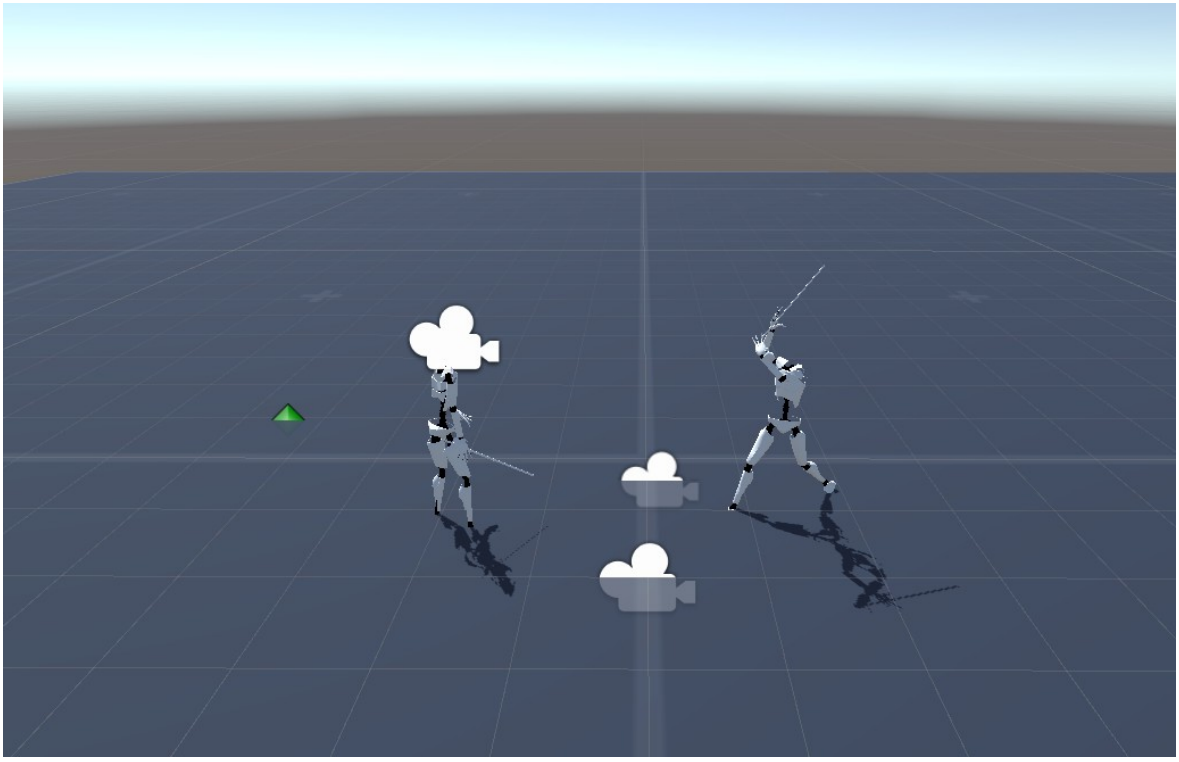
Görsel 3.5: Direct Play lobisi, Axis Neuron'un IP adresi girilmiş, yalnızca bir kullanıcı mevcut.

Lobinin işlevsel kısmını oluşturan LobbyManager objesi üzerinde bir Lobby Manager betiği mevcuttur. Bu betiğin parametreleri modifiye edilerek simulasyon başladığı zaman kullanıcılar için simulasyon ortamına çağırılacak model olarak istenen modelin prefab'i verilebilir ve simulasyon ortamı olarak da istenen sahne sunulabilir. Kullanıcıların girdiği IP adreslerinde yayın yapan Axis Neuron'a bağlanabilmeleri için bu LobbyManager objesine NetworkLobbyHook isimli bir betik eklenmiştir. Bu betik içerisinde, lobi sahnesinden simulasyon ortamına geçiş yaparken çağırılacak modeller/prefab'ler üzerinde, lobi sırasında mevcut bilgiler kullanılarak değişiklik yapılabilmektedir. Betik, lobi ekranında girilen IP adreslerinin alınıp, kullanıcı avatarını oluşturan objenin/modelin üzerindeki NeuronAnimatorInstance betiğindeki Address değişkeninin üzerine yazmak için kullanılmaktadır. Bu sayede kullanıcılar simulasyona girdikleri zaman avatarları, lobideki girdikleri adrese yayın yapan Axis Neuron'dan alınan hareketleri izlemektedir.

3.2.2 Simulasyon Çevresi

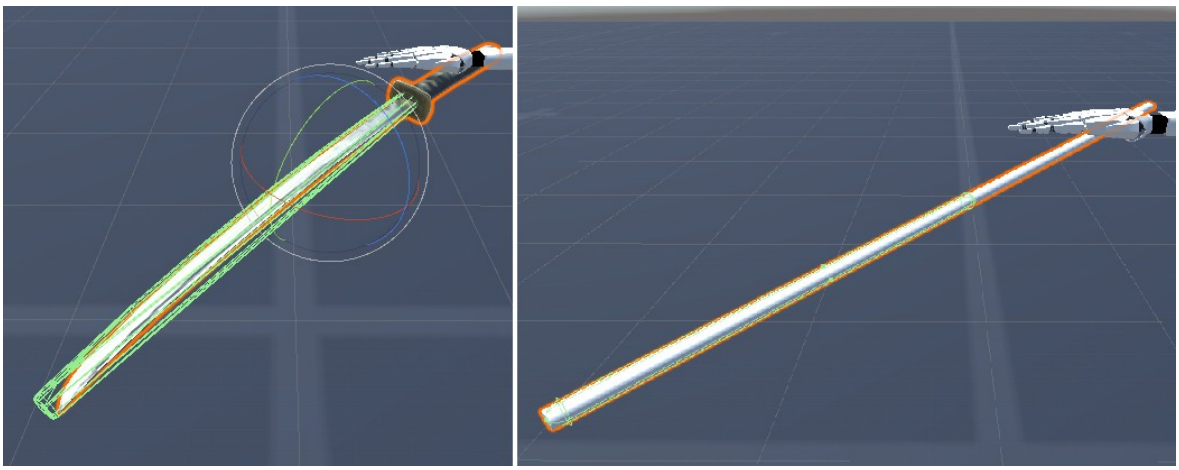
Simulasyon alanı, boş bir düzlemden oluşmaktadır. Bu düzlemde kullanıcılar bir başlangıç noktasında alana girdikten sonra, kendi avatarlarını vücut hareketlerini takip edecek şekilde yöndendirebilir ve birbirleri ile etkileşime geçebilirler. İki avatar mevcutken simulasyon alanının dışarıdan görüntüsü Görsel 3.6'da görüldüğü gibidir. Simulasyon alanında harici bir nesne bulunmamasına karşın, eklenmesi durumunda kullanıcılar çevrelerindeki nesneler ile fiziksel etkileşime(itmek, çarpmak vs.) geçebilmektedirler. Kullanıcı avatarının gerçeklenmesi için, Perception Neuron Unity SDK'sı tarafından sağlanan sanal gerçeklik uygulamaları için ayarlanmış NeuronRobot_MultiMesh isimli bir insan modeli prefab'i kullanılmıştır. Bu prefab üzerinde yapılan kimi değişiklik ve eklemeler sayesinde çok kullanıcılı bir sanal gerçeklik ortamında üzerindeki kameranın sıkıntısız çalışması sağlanmış, çevre nesnelerle fiziksel etkileşime girmesi mümkün kılınmış ve simulasyon mekanikleri gerçeklenmiş ve avatarın eline simulasyon sırasında

kullanabileceği bir kılıç eklenmiştir. Kamera, fiziksel etkileşim ve oyun mekaniklerinin nasıl çalıştığı ilerleyen kısımlarda daha ayrıntılı açıklanacaktır.



Görsel 3.6: İki avatar ile simulasyon alanı görüntüsü, Unity içerisinde uygulama çalışırken scene ekranından alınmıştır.

Avatarın elindeki silah için birden fazla alternatif çözüm denenmiştir. Projenin güncel haline kullanılan silah, Unity içerisinde tasarlanmış basit bir eliptik silindirdir ve keskin olması gereken kısmında kapsül şeklinde bir collider Görsel 3.7'de görüldüğü gibi mevcuttur. Görsel 3.7'de görülen diğer silah ise, Unity Asset Store'dan edinilmiş ücretsiz bir assetin modifiye edilmiş halidir. Bu silahın çelik kısmı, karşıdaki kullanıcının avatarına temas ettiği takdirde karşıdaki kullanıcının avatarının ölmesine sebep olması beklenmektedir. Bu sebeple çarpışmaları algılaması için üzerine bir mesh collider ve fizik kuvvetlerinden etkilenmesi için bir rigidbody eklenmesi gerekmiştir. Ancak, kılıcı tutan avatarın kendisini öldürmesine engel olmak için mesh collider'ın yalnızca kılıcın çelik kısmını kapsaması gerekmektedir. Bu amaçla kılıca, tamamını kaplayan bir mesh collider yerine, kabzası için tasarlanmış ve yalnızca çelik kısmın etrafını kapsayan bir mesh collider eklenmiştir. Rigidbody'nin düzgün çalışabilmesi için bu mesh collider convex olarak ayarlanmıştır. Bu silahın şu anda kullanılmamasının sebebi ise rigidbody ile ilgili bir problemden kaynaklı olarak simulasyon başladığında silahın aşağı düşmesidir. Bu silahın rigidbody'li ve rigidbody'siz hallerini kullanan avatarlar da farklı prefab'ler olarak proje dosyaları içerisinde mevcuttur (NeuronRobot_MultiMesh_sword_rb ve NeuronRobot_MultiMesh_sword_norb).

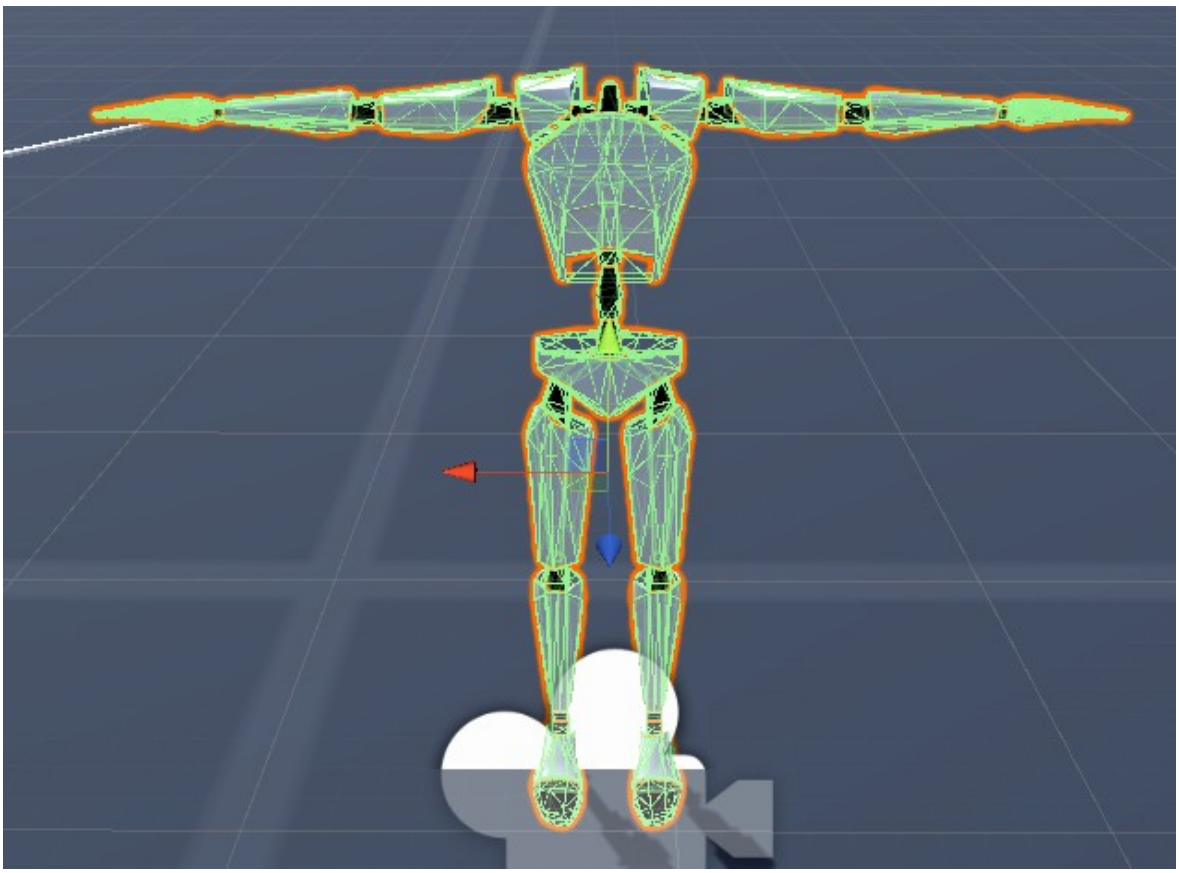


Görsel 3.7: Solda: Asset Store'dan edinilip modifiye edilen kılıç. Sağda: Unity içerisinde tasarlanan şu an kullanılan silah. İki görselde de collider'lar yeşil çizgilerle gösterilmiştir.

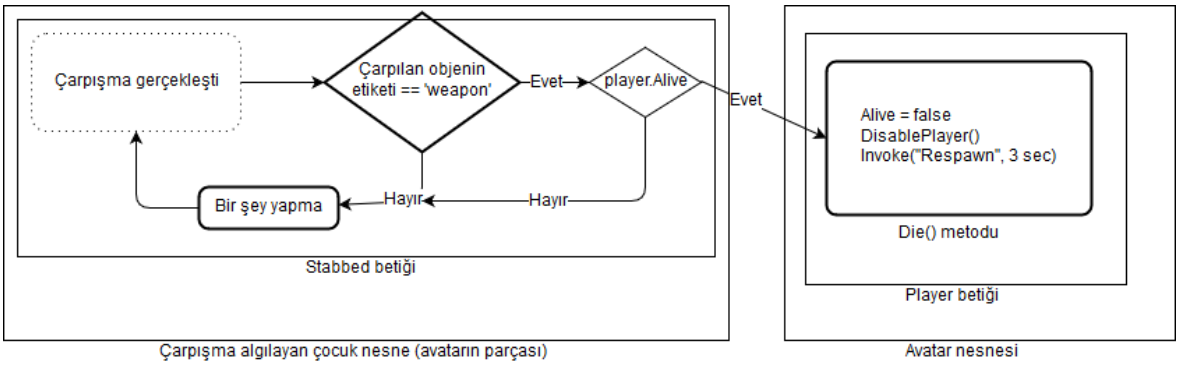
Bu şekilde avatarın eline sabitlenmiş bir silah kullanılmasının sebebi, Perception Neuron'un el ve parmak hareketlerini algılamadaki hassasiyetinin ve Unity fizik motorunun çok küçük kuvvet değişimlerini hesaplama kabiliyetinin söz konusu silahı kullanıcının hareket yakalama aracığı ile elinde tutabilmesi için yetersiz olmasıdır. Projenin ilerleyen iterasyonlarında, kullanıcının kılıcı iki eli ile daha yüksek hareket özgürlüğü ile tutabilmesi için farklı mekanizmalar geliştirilecektir.

3.2.3 Çarpışmalar ve Fizik Motoru

Perception Neuron Unity SDK'sı, insan modellerine rigidbody eklemek için Skeleton Tools isimli bir araç sunmaktadır. Ancak SDK'nın güncel sürümlerinde aynı aracı collider eklemek için kullanmak mümkün değildir. Proje boyunca bu konuda doküman eksikliği sebebiyle sıkıntılar yaşanmıştır. Sonunda sorun, insan modeli objesinin uygun çocuk objelerine convex collider'ların Görsel 3.8'de görüldüğü gibi tek tek eklenmesi ile çözülmüş ve kullanıcı avatarının çevresindeki nesneleri manipule edebilmesi mümkün kılınmıştır. Bunun bir sonraki aşaması ise, çarpışmaların simulasyon içerisindeki kılıç temasında ölme gibi mekaniklerin işlenebilmesi için kullanılması olmuştur. Bu aşamada yaşanan önemli bir sorun, Unity'nin çarpışma algılama mekanizması sebebiyle collider'lar tarafından algılanan çarpışmaların obje hiyerarşisinde içinde rigidbody barındıran en yakın üst(parent) nesne tarafından işlenmesi sebebiyle ortaya çıkmıştır. Bu çarpışmaların algılanıp işlenebilmesi için, içinde rigidbody bulunan yani çarpışma verisininin iletildiği tüm nesnelerin içerisine çarpışmaları algılayacak Stabbed isimli bir betik eklenmiştir. Kullanıcıların tüm eklemleri, "Player" etiketine sahip nesnelerdir; ellerindeki kılıçlar ise "weapon" etiketine sahiptir. Çarpışma algılama betiği, kendi etiketi Player ise ve çarptığı nesnenin etiketi weapon ise, kendisinin bağlı olduğu kök nesneye bağlı Player betiğindeki Die() metodunu çağırarak bağlı olduğu oyuncu avatarının ölmesini sağlar. Player betiğinin içerisindeki Alive isimli bir boolean değer avatarın canlı olup olmadığını takip ederek, birden fazla eklemle çarpışma halinde Die() metodunun birden çok kez çağrılmasına engel olur. Görsel 3.9'daki şema da bu mekanizmayı açıklamaktadır.



Görsel 3.8: Üzerindeki collider barındıran tüm objeler seçili avatar, seçili objelerin colliderları yeşil çizgiler ile gösterilmiştir.

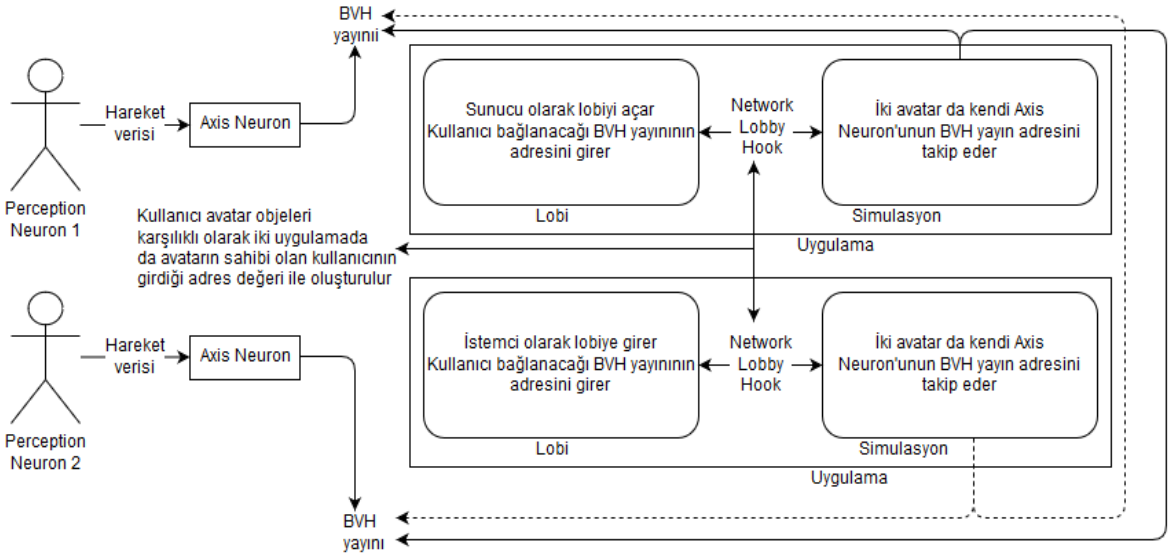


Görsel 3.9: Çarpışmaların çalışma mekanizması

3.2.4 Ağ Üzerinden Kullanıcıların Bağlanması

Uygulamada ağ kullanıcılar arası ağ bağlantısı kurma amaçlı Unity'nin sağladığı Multiplayer High Level API ve diğer ağ özelliklerinden faydalanılmıştır. Projenin başlarında, oyuncular arası bağlantının tamamiyle bu yapı üzerinden, bir sunucu-istemci yapısı kullanılarak gerçeklenmesi planlanmıştır. Ancak ilerleyen kısımlarda bu yapıdan vaz geçilip, bunun yerine yalnızca hareket verilerinin aktarıldığı, bunun da Unity'nin sunduğu ağ özelliklerinden faydalanmak yerine Axis Neuron'lardan yayınlanan hareket verilerine iki kullanıcının da direkt olarak kendi yerel makinaları üzerinden bağlandığı bir sistem

kullanılmıştır. Uygulama içerisinde Unity'nin ağ özelliklerinden yalnızca Lobi sahnesinde ve simulasyon içerisinde kullanıcıların doğma ve yeniden doğma süreçlerinde faydalanılmıştır. Bunun haricinde uygulamanın geri kalan kısımları, asenkron biçimde kullanıcıların kendi lokal makinalarında çalışmaktadır. Bu çalışma prensibi Görsel 3.10'da basitçe aktarılmıştır.



Görsel 3.10: Kullanıcı hareketlerinin aktarılması. Görüldüğü üzere iki uygulama da iki kullanıcının avatarlarına ait BVH verisini direkt olarak o kullanıcıya ait Axis Neuron'dan almaktadırlar.

Tam senkronize bir çoklu kullanıcı yapısı yerine bu şekilde sadece hareket verisinin aktarıldığı ve tüm hesaplamaların lokal makinalarda yapıldığı bir yapı tercih edilmesinin altında üç temel motivasyon yatmaktadır: Birincisi, tüm eklemlerinin hareket verisini bir hareket yakalama cihazının yayınladığı verilerden alan ve gerçekçi fizik özelliklerine sahip olması gereken bir avatarın tüm eklemlerinin hareketlerini ağ üzerinden aktarmak mümkün ancak karmaşık bir işlemdir. İkincisi, Bu şekilde hareket verisinin Axis Neuron'dan yazılan uygulamaya, oradan da işlenip uygun biçimde ağ üzerinden uzaktaki makinaya aktarılması ve sunucu-istemci arasındaki istek-yanıt süreçleri uygulamaya çok ciddi gecikme süreleri ekleyip, bir simulasyonda olması gereken gerçekçiliği elde etmeyi zorlaştıracaktır. Üçüncüsü de, her ne kadar kullanıcıların uygulamayı asenkron biçimde kendi makinalarında çalıştırıyor olmaları lokal modifikasyonlara ve hile yapmaya çok ciddi olanak sunsa da, unutulmamalıdır ki gerçeklenmeye çalışılan uygulama bir oyun değil, simulasyondur. Rekabetçi bir puan sistemi bulunmamaktadır ve amaç, rakibi yenmek değil, karşılıklı çalışmaktır. Dolayısıyla kullanıcıların hile yapmasına imkan sağlanması aslında önemli bir problem değildir, gecikmeleri düşürüp gerçekçiliği korumak çok daha önemli bir kıstas.

Perception Neuron'dan alınan verilerin ağ üzerinden aktarılması, iki kullanıcının, uygulamanın iki cihazdaki asenkron çalışan örneklerinin ikisinde de avatarlarına bağlı olan NeuronAnimatorInstance betiklerindeki address değişkeninin kendi Axis Neuron'larının yayın yaptığı adresi tutması ile mümkün kılınmıştır. Bu da yukarıda Lobi bölümünde de anlatıldığı gibi, Lobi ekranında girilen adresin çekilip simulasyon ekranına geçiş aşamasında oluşan kullanıcıya ait avatara aktarılması ile sağlanmıştır. Söz konusu betik Perception Neuron Unity SDK'sı tarafından sağlanmış olmasına karşın, bu şekilde çalışabilmesi (iki kullanıcı için ayrı address değerleri tutabilmesi) için bu script içerisinde tanımlanan NeuronAnimatorInstance sınıfının temel aldığı (kalıtıldığı/inherit edildiği)

NeuronInstance sınıfının bulunduğu betikte kimi değişiklikler yapılması gerekmiştir. Sınıf, MonoBehaviour yerine NetworkBehaviour sınıfını temel alacak şekilde düzeltilip, address değişkeni ve kimi başka değişkenler SyncVar olarak yeniden tanımlanmıştır.

Bunun haricinde ağ kullanan özellikler, kullanıcı avatarlarına ait mekaniklerin tanımlandığı Player isimli betik içerisindeki başlama ve ölünce yeniden doğmayı kontrol eden Start() ve Respawn() metotları ile kameraların aktive edildiği EnablePlayer() metodlarıdır. Bu metotlarda kullanıcının sunucu veya istemci olmasına bağlı olarak (her ne kadar tam olarak bir sunucu-istemci sistemi kullanılmasa da Unity için kullanıcılardan biri sunucu diğeri istemcidir) simulasyon alanında doğacakları nokta belirlenir ve iki avatardan sadece o anda kontrol edilenin kamerası aktifleştirilir.

3.2.5 Simulasyon Akışı ve Mekanikler

Bu kısımda simulasyon sahnesine geçildikten sonra uygulamanın çalışma akışı ve mekanikleri açıklanmıştır.

Simulasyonun kullanıcılar için akışını, iki kullanıcının da avatarına eklenmiş olan Player isimli betik belirlemektedir. Betiğin nasıl çalıştığı Görsel 3.11'deki akış diyagramında da gösterilmiştir.

Bu betik içerisindeki Start() metodu başlangıcı belirler. Öncelikle iki kullanıcı da önceden belirlenmiş sabit bir konumda doğacak şekilde ayarlanırlar: Sunucu için (0, 0, 0) ve istemci için (0, 0, 2). Ardından betiğin bağlı olduğu kullanıcı avatarını çalıştıracak EnablePlayer() metodu çağırılır.

EnablePlayer() metodu içerisinde, avatarı aktifleştirmek için öncelikle kullanıcının görüşünü sağlayacak, avatarın kafası yerine yerleştirilmiş kamera aktive edilir. Ardından avatara bağlı olan NeuronAnimatorInstance betiği aktive edilir ve Axis Neuron ile bağlantısı kurulur.

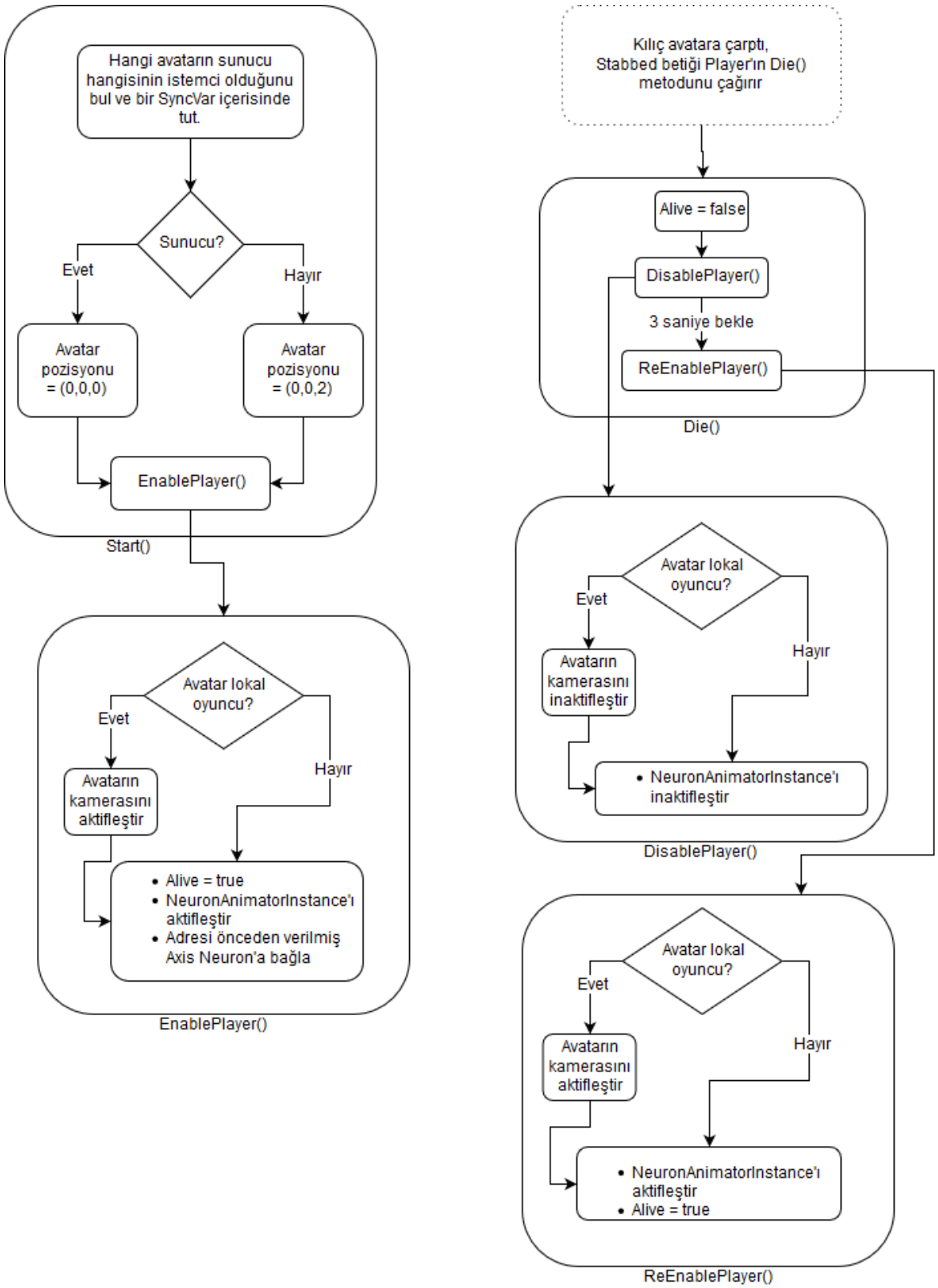
Simulasyon bu şekilde başlar ve kullanıcılar avatarlarını kontrol ederek ellerindeki kılıçlar yardımıyla birbirlerine karşı hamle yapabilirler. Kılıçların oyunculara temas edip etmediği ise, üzerinde rigidbody bulunan tüm vücut parçalarına eklenmiş olan Stabbed betiği ile kontrol edilir. Simulasyon içerisinde yalnızca kılıçlar "weapon" olarak etiketlenmiştir, oyuncu avatarlarının tüm parçaları ise "Player" olarak etiketlenmiştir. Stabbed scripti, bağlı olduğu nesne "Player" etiketine ve çarptığı nesne "weapon" etiketine sahipse çalışır, ve kendisinin bağlı olduğu nesnenin en üst atası (root) olan kullanıcıya bağlı Player betiği içerisinde yer alan Die() metodunu çağırarak çarpışmanın gerçekleştiği avatarın ölmesini sağlar. Birden fazla eklemin çok kısa aralıklarla çarpması durumunda Die metodunun birden çok kez çağrılmasını önlemek için avatarın hayatta olup olmadığını takip eden bir boolean değişken tutulmaktadır. Kılıcın sapı ile temasın silahın sahibi olan avatarı öldürmesine engel olmak için de kılıcın mesh collider'ı olarak kılıcın tamamını kapsayan kendi collider'ı yerine yalnızca çelik kısmın etrafını kapsayan kabza için tasarlanmış collider kullanılmıştır.

Avatarı öldüren Die() metodu ise, avatarı kullanılmaz hale getirmek için DisablePlayer() metodunu çağırdıktan sonra, 3 saniye içerisinde yeniden doğması için Respawn() metodunu gecikmeli olarak çağırır.

DisablePlayer() metodunun ise tüm yaptığı, kullanıcını kamerasını ve NeuronAnimatorInstance betiğini deaktive ederek ekranının kararmasını sağlamak ve avatarını kontrol etmesine engel olmaktır.

Bunun ardından çağırılan Respawn() metodu, kullanıcının simulasyona başlama pozisyonunu tekrar ayarlar ve ReEnablePlayer() metodunu çağırır.

Son olarak da ReEnablePlayer() metodu, simulasyonun başlangıcı için yapılması gereken bazı ekstra adımları atlamak şartıyla EnablePlayer() metodunun yaptıklarının birebir aynısını yapar: kullanıcı kamerası ve NeuronAnimatorInstance betiğini yeniden aktifleştirir.



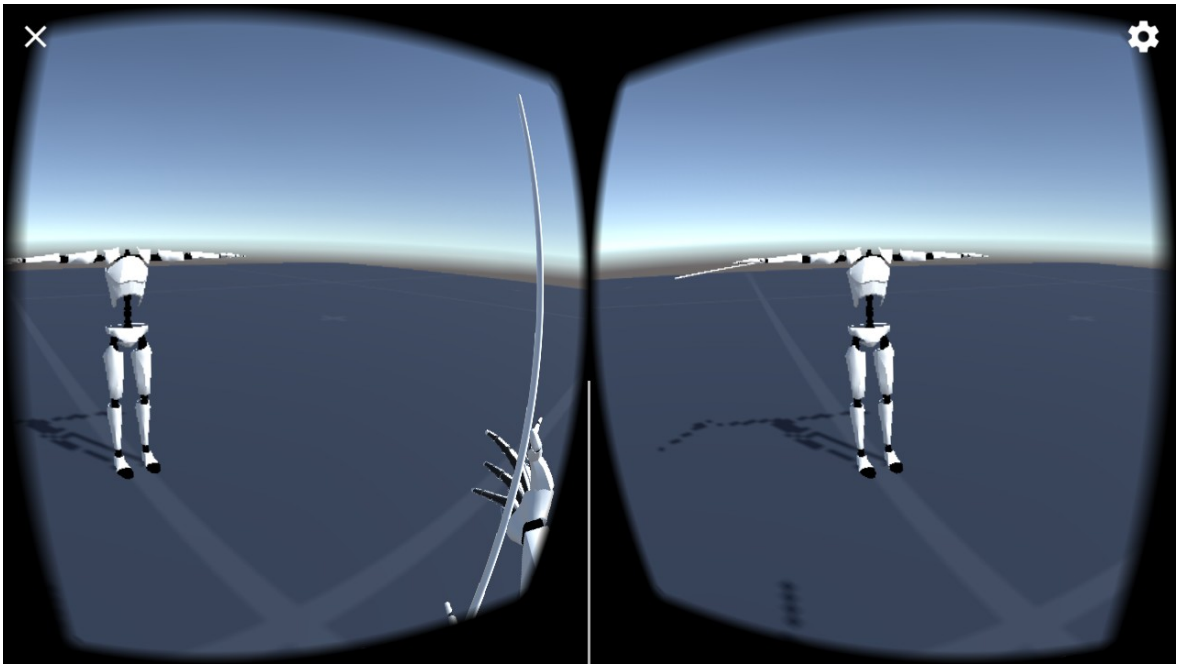
Görsel 3.11: Player betiği için akış diyagramı

3.2.6 Sanal Gerçeklik

Sanal gerçeklik kısmının gerçeklenmesi için Google Cardboard SDK'sından, Unity'nin XR API'sinden ve Perception Neuron Unity SDK'sının içinde bulunan sanal gerçeklik örneklerinden faydalanılmıştır. Uygulama ilk açıldığı sırada kullanıcının karşısına çıkan lobi ekranı sanal gerçeklik kullanmamaktadır. Bunun bir numaralı sebebi, sanal gerçeklik kullanırken IP adresi girmenin kullanıcı açısından yaratacağı zorluktur. Simulasyon başladığında ise, kullanıcı telefonunun ekranına çift tıklayarak sanal gerçekliği açabilir veya kapatabilir.

Sanal gerçeklik için avatarın kafasının yerine kamera yerleştirmek gerekmiştir. Avatarın kafasının olmamasının sebebi de budur. Bu kamera, Camera Holder isimli bir GameObject'in çocuk objesi olarak ayarlanmıştır. Camera Holder ise, yukarıda bahsedilen sanal gerçeklik örneklerinde kullanılan Neuron VR Adapter isimli bir betiğe sahiptir. Bu betik, kameranın bakış açısı olarak telefonun yönelişini baz almasını, konum olarak ise avatarın kafasının olması gereken bağlantı noktasını takip etmesini sağlar. Örnekten farklı olarak, bu projede Camera Holder bağımsız bir nesne olarak değil, avatarın bir çocuk objesi olarak gerçeklenmiştir. Bunun sebebi, avatarın önceden simulasyon alanında bulunan değil, doğan (spawn olan) bir nesne olmasıdır.

Çift tıklama halinde sanal gerçekliğe geçme mekanizması için ise Camera Holder'ın çocuk objesi olan kameraya VR Toggle isimli bir betik eklenmiştir. Bu betik basitçe, çift tıklama algıladığı zaman kullanılan Unity XR'ın kullandığı sanal gerçeklik SDK'sını "cardboard"dan "None"a ("hiçbiri"ne) çevirmektedir. Sanal gerçeklik moduna geçildiğinde oyun içi bir görüntü Görsel 3.12'de görülebilir.



Görsel 3.12: Sanal gerçeklikle oyun içi bir görüntü. Karşıda görünen avatar herhangi bir Axis Neuron'a bağlanmamış.

4 Test Ortamı ve Testler

Bu proje için yapılması planlanan testler, test ekipmanı (ikinci Perception Neuron) eksikliği ve zaman darlığı sebepleriyle tamamlanamamıştır. Tek Perception Neuron ve kaydedilmiş hareketleri oynatan ikinci bir Axis Neuron yazılımı ile temel fonksiyonların çalışıp çalışmadığını anlamak için bir takım testler yapılmıştır.

4.1 Test Ortamı

Planlanan testlerde test ortamının şu şekilde olması beklenmekteydi: İki adet Perception Neuron, her birine Wi-Fi ile bağlı Axis Neuron çalıştıran birer bilgisayar, bu Axis Neuron'lara bağlanan iki adet Android telefon, bu telefonlarla kullanılmak üzere sanal gerçeklik gözlükleri ve yeterli hareket özgürlüğünü sağlayacak kadar açık alan. Testler daha önce savaş sanatları üzerine tecrübesi olan ve olmayan kullanıcıların ayrıca değerlendirmelerini toplayacak şekilde planlanmıştı.

Şu ana kadar tek cihazla yapılması mümkün olmuş testlerin kapsamı ise performans değerlendirmesinden ziyade hatların tespiti üzerine olmuştur. Kılıçla yapılan basit bir kesiş tekniği önceden kaydedilmiş ve bir Axis Neuron'da oynatılmıştır. Sunucu, hata ayıklamayı kolaylaştırmak adına bir dizüstü bilgisayarda çalışan Unity içerisinde çalıştırılmıştır. Perception Neuron, Cardboard ve bir Android telefon kullanan tek bir kullanıcı ise bu sürekli aynı kesiş hareketini yapan "kullanıcı"nın karşısına geçmiştir.

4.2 Testler

Yapılması planlanan testler, savaş sanatlarında farklı düzeyde tecrübeye sahip insanlara uygulamayı kullandırıp; kullanım kolaylığı, eğitime getirebileceği kolaylıklar, geçikme ve hataların ne kadar dikkat çektiği gibi çeşitli konularda deneklere anket yapmak şeklindeydi. Bu testler de en yakın zamanda gerçekleştirileektir.

Şu ana dek gerçekleştirilen tek kullanıcılı testlerde ise yazılımın genel çalışmasında hata olup olmadığı kontrol edilmiştir. Sistemin güvenilirliğinin şu aşamada tatmin edici olmadığı söylenebilir. İstikrarlı olmayan, çok defa çalıştırıldığında bir kaç defa rastlanan bazı hatalar mevcut. Bu hatalar genellikle Axis Neuron ile avatarlardan birinin ilk bağlantıyı kuramaması şeklinde göstermektedir kendisini. Hataların etrafından dolanmak için kimi yöntemler geliştirilmiş olmakla birlikte, daha kapsamlı testler yapıp daha geniş kapsamlı bir hata ayıklama sürecinin yürütülmesinin ciddi miktarda faydası olacaktır.

Yapılan testlerde simulasyonda kullanıcının fark edebileceği düzeyde bir gecikme tespit edilmemiştir. Simulasyonun iki ayrı cihazda çalışan örnekleri yapılan testlerde tamamen eş zamanlı hareket etmişlerdir.

Kullanılabilirlik konusunda ise kimi sıkıntılar olduğu aşikardır. Kılıcın ele sabitlenmiş olması, özellikle eldeki Neuron sensörünün kalibrasyonunda bir kayma olduğu zaman kılıcın kontrolünü zorlaştırmaktadır. Bununla ilgili çözüm önerileri ileride tartışılacaktır.

5 Kıyaslamalı Değerlendirme ve Tartışma

Eldeki test sonuçlarının yetersizliğinden ötürü, kıyaslamalı bir değerlendirme yapacak pek fazla nicel veri bulunmamaktadır. Ancak, bu projenin ilham kaynaklarından biri olan, Hareket Yakalama ile Sanal Gerçeklikte Tenis Oyunu [5] projesi ile nitel bir kıyaslama yapmak mümkündür. Sözü edilen projede, tek kişilik bir oyun alanında sanal gerçeklik kullanılarak tek kişilik bir tenis oyunu simulasyonu gerçeklenmiştir. Bu yeni projenin bunun üzerine koyduğu en önemli unsur, ağ üzerinden bağlı birden fazla kullanıcının varlığıdır. Bu, daha ilerleyen çalışamalarda, rekabetçi oyunlardan birlikte yapılması gerekecek fiziksel aktivitelere çok sayıda uygulamaya alan açmaktadır.

6 Sonuç ve Gelecek Çalışmalar

Bu projede iki kişinin ayrı konumlardan bağlanıp karşılıklı olarak çalışabileceği bir savaş sanatları simulasyonu yapılması amaçlanmıştır. Şimdilik, projenin kapsamı genel bir savaş sanatları simulasyonundan ziyade, kılıç dövüşü ile kısıtlı tutulmuştur. Amaç, iki kişinin karşılıklı olarak gerçekçi bir simulasyon ortamının içerisinde normal koşullarda gerçek silahlarla çalışamayacakları biçimde kılıçla mücadele tekniklerini çalışabilmeleridir. Sonuç olarak elde edilen üründe bu amaç gerçeklenmiştir. Elbette düzeltilebilecek, ek yapılabilecek, eniyilenebilecek çok sayıda noktası olmakla beraber; bütün bunların üzerine inşa edilebileceği sağlam bir temel oluşturulmuştur.

Bu noktada artık, elde edilen uygulamanın hangi eksiklerinin kapatılabileceği ve üzerine nasıl gelişmeler yapılabileceğini tartışmak gerekir.

İlk olarak, yeterli miktarda test yapılmadığından daha önce bahsedilmişti. Dördüncü bölümde tanımlanan test ortamının kurulması ve gerekli testlerin gerçekleştirilmesi, projeyi hızlı bir hata ayıklama ve eniyileme sürecine sokacaktır.

Kullanıcı deneyimini arttırmak için ise bu projeye yapılabilecek kimi eklemelerden burada bahsetmek gerekiyor. Öncelikle, kullanılabilirliği ciddi anlamda düşüren silahın ele sabitlenmesi problemi için bir çözüm önerisi: kılıcın sapını iki ele bağlı eksenlar etrafında döner hale getirmek. Aikidoda, kılıç kullanımının esaslarından birisi kılıcı sabit bir biçimde tutmak yerine, bir kaldıraç misali bir eli destek noktası diğer eli ise kuvvet kolu yaparak hamle yapmaktır. Bu prensip bu projeye uygulanabilir, kılıcın sağ elde bir eksen etrafında dönmesi sol elin ise bu dönüşü kontrol etmesi şeklinde bunu gerçeklemek mümkün.

İkinci bir kullanıcı deneyimi arttırıcı iyileştirme de muhakkak silah diyazını ile ilgili olacaktır. Ucundaki belirli bir kısmı keskin kabul edilip dokunduğu gibi öldüren bir silah yerine, gerçekçi bir kılıç dizaynı yapılıp çeliğin sadece keskin kısmıyla temasların öldürmesi sağlanabilir. Hatta daha da iyisi, kılıcın yalnızca belirli bir hızla çarptığı hedefleri öldürmesi sağlanabilir.

Bunların haricinde yapılabilecek önemli bir ekleme ise, kapsamı kılıç dövüşünün dışına çıkarıp, yumruk tekme gibi saldırıların ve bunlara karşı blok alma mekanizmalarının gerçeklenmesi olacaktır. Burada temas halinde direkt ölmekten ziyade, bir puan veya kalan can sistemi kullanılabilir.

Son olarak akla gelen olası geliştirmeler, daha iyi (mümkünse sanal gerçeklikle uyumlu çalışan) bir lobi ekranı ve kimi kozmetik iyileştirmeler olacaktır.

Burada sayılan yapılabilecek düzeltme ve geliştirmelerin hepsi bu projenin devamı için yakın zamanda uygulanması planlanan iyileştirmelerdir.

7 Referanslar

[1] Y. Ye, S. Ci, A. K. Katsaggelos, Y. Liu, "A multi-camera motion capture system for remote healthcare monitoring" in *2013 IEEE International Conference on Multimedia and Expo (ICME), San Jose, CA, USA, July 15-19, 2013*.
[2] A. L. Rincon, H. Yamasaki, S. Shimoda, "Design of a video game for rehabilitation using motion capture, EMG analysis and virtual reality" in *2016 International Conference on Electronics, Communications and Computers (CONIELECOMP), Cholula, Mexico, February 24-26, 2016*.
[3] J. Chan, H. Leung, J. K. T. Tang, T. Komura, "A Virtual Reality Dance Training System Using Motion Capture Technology," *IEEE Transactions on Learning Technologies*, vol. 4, no. 2, pp. 187-195, April-June 2011. [Online]. Available: IEEE Xplore, https://ieeexplore.ieee.org/document/5557840. [Erişim: January 4, 2019].
[4] S. Royston, C. DeFanti, K. Perlin, "A Collaborative Untethered Virtual Reality Environment for Interactive Social Network Visualization," arXiv: arXiv:1604.08239 [cs.HC], Apr. 2016.
[5] E. Bilgin, "Hareket Yakalama ile Sanal Gerçeklikte Tenis Oyunu", İstanbul Teknik Üniversitesi, 2018
[6] "Unity – Manual: Asset Workflow," https://docs.unity3d.com/Manual/AssetWorkflow.html [Erişim: 4 Ocak 2019]
[7] "Unity – Manual: Colliders," https://docs.unity3d.com/Manual/CollidersOverview.html [Erişim: 4 Ocak 2019]
[8] "Unity – Manual: Using Components," https://docs.unity3d.com/Manual/UsingComponents.html [Erişim: 4 Ocak 2019]
[9] "Unity – Manual: GameObject," https://docs.unity3d.com/Manual/class-GameObject.html [Erişim: 4 Ocak 2019]
[10] "Polygon Mesh – Wikipedia," https://en.wikipedia.org/wiki/Polygon_mesh [Erişim: 4 Ocak 2019]
[11] "Unity – Scripting API: MonoBehaviour," https://docs.unity3d.com/ScriptReference/MonoBehaviour.html [Erişim: 4 Ocak 2019]
[12] "Unity – Manual: Prefabs," https://docs.unity3d.com/Manual/Prefabs.html [Erişim: 4 Ocak 2019]
[13] "Unity – Manual: Rigidbody," https://docs.unity3d.com/Manual/class-Rigidbody.html [Erişim: 4 Ocak 2019]
[14] "Unity – Scripting API: SyncVarAttribute," https://docs.unity3d.com/ScriptReference/Networking.SyncVarAttribute.html [Erişim: 4 Ocak 2019]
[15] "MOCAP 101 | Perception Neuron by Notiom," https://neuronmocap.com/content/mocap-101 [Erişim: 4 Ocak 2019]
[16] "Unity (game engine) – Wikipedia," https://en.wikipedia.org/wiki/Unity_(game_engine) [Erişim: 4 Ocak 2019]