AFYON KOCATEPE ÜNİVERSİTESİ
AKADEMİK MÜZİK ARAŞTIRMALARI DERGİSİ

Cilt VIII / ÖZEL SAYI / Ekim 2022
Müzikte "YENİ"
E ISSN: 2667-6001

SANAL GERÇEKLİK İÇİN NESNE ODAKLI, YORDAMSAL ve ETKİLEŞİM TEMELLİ SES-MÜZİK TASARIMI

Object-Oriented, Procedural and Interaction Based Sound-Music Design for Virtual Reality

DOI NO: 10.36442/AMADER.2022.71

Cihan IŞIKHAN[*]
Arda EDEN[†]

Özet

Son yıllarda Sanal Gerçeklik (SG) uygulamaları oldukça yaygınlaştı ve gelişti. Diğer alanlarda olduğu gibi ses ve müzik de bu durumdan yakından etkilendi. Özellikle Covid-19 salgını sırası ve sonrasında artan çok amaçlı VR çalışmaları, ses ve müzik üretim/tasarım modellerini eskiye göre çok daha gerçekçi olarak sanal ortam içine yerleştirmiş durumda. Bunda kuşkusuz cyberspace'den metaverse'e kadar geçmişten günümüze tüm sanal ortam ve çalışmalarının katkısı büyük ve bu çalışmalar tüm hızıyla halen sürüyor. Özellikle oyun sektörü, ses ve müziği içine alan çoklu ortam alanlarının hemen hepsine kılavuzluk eder durumda.

Bu çalışmada, SG özelinde günümüz sanal ortam teknik ve yöntemlerini içeren çalışmalarla, ilgili temel terim ve kavramlar anlatılarak; sanal ortam elektronik müzik performansı bağlamında nesnelerin etkileşimli bir yordamsal ses tasarım süreci ile ne şekilde ilişkilendirilebileceğine yönelik yöntemler değerlendirilmiş, bu yöntemlerin bir arada kullanıldığı sanal bir performans mekânı tasarlanmıştır. Kullanıcı, bir SG gözlüğü ve kontrolcüler aracılığıyla Etkileşimli Müzik Odası[‡] olarak adlandırılan sanal bir mekânda gezinebilmekte ve odada yer alan iki elektronik çalgı ile etkileşerek ses üretebilmektedir. Yordamsal ses üretim tekniğinin ön plana çıktığı bu uygulama, Unity oyun motoru ve Chunity eklentisi kullanılarak hazırlanmıştır.

Anahtar Kelimeler: *Müzik Teknolojisi, Sanal Gerçeklik, Yordamsal Ses, Ses-Müzik Tasarımı.*

Abstract

In recent years, Virtual Reality (VR) applications have become quite widespread and developed. As in other fields, sound and music are closely affected by this situation. Sound and music production/design models have been placed in multi-purpose VR studies much more realistic than before, which increased especially during and after the Covid-19 pandemic. Undoubtedly, all VR studies from past to present, from cyberspace to metaverse, have significantly contributed, and these studies are continuing at full speed. In particular, the gaming industry guides almost all the multimedia fields, including sound and music.

In this study, the basic terms and concepts related to VR studies involving up-to-date virtual environment techniques and methods are explained. Then, the methods regarding the interaction between objects and the interactive procedural sound design process in the context of electronic music performance in VR are evaluated, and a virtual performance space where these methods are used together is designed. The user can navigate in this space called the Interactive Music Room[‡], using VR goggles and controllers, and can produce sound by interacting with two electronic instruments in the room. The application, in which the procedural sound production technique comes to the fore, is implemented using the Unity game engine and Chunity plugin.

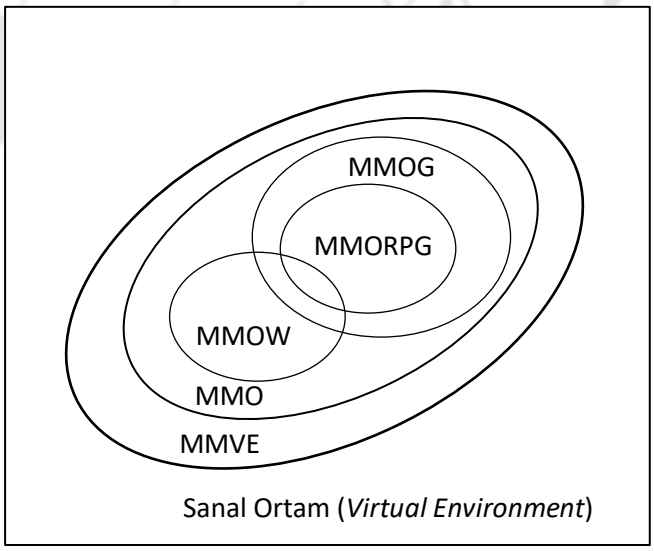
Keywords: *Music Technology, Virtual Reality, Procedural Audio, Sound-Music Design.*

[*] Prof. Dr., Dokuz Eylül Üniversitesi Güzel Sanatlar Fakültesi Müzikoloji Bölümü Müzik Teknolojisi Anabilim Dalı, cihan.isikhan@deu.edu.tr
[†] Prof. Dr., Yıldız Teknik Üniversitesi Sanat Tasarım Fakültesi Müzik ve Sahne Sanatları Bölümü, ardaeden@yildiz.edu.tr
[‡] Sanal gerçeklik ortamını da içeren Etkileşimli Müzik Odası, İstanbul Kalkınma Ajansı destekli TR/10/12/YES/0028 numaralı proje kapsamında Yıldız Teknik Üniversitesi Sanat ve Tasarım Fakültesi'nde kurulan IMC Laboratuvarının teknik altyapısı kullanılarak hazırlanmıştır.

Giriş

Son yıllarda Sanal Gerçeklik (SG) uygulamaları oldukça yaygınlaştı ve gelişti. Diğer alanlarda olduğu gibi ses ve müzik de bu durumdan yakından etkilendi. Özellikle Facebook kurucusu Zuckerberg'in yakın zamanda duyurduğu Meta şirketiyle birlikte artan *metaverse* uygulamaları, Sony gibi dünya devi müzik prodüksiyon şirketlerini de harekete geçirdi. Tüm bu gelişmeler, uzun zamandan bu yana bilgisayar bilimleri başta olmak üzere ilgili birçok alanda akademik/mesleki olarak tartışılan bir kavramı tekrar gündeme getirmiş oldu: Sanal Ortam (*Virtual Environment*).

Singhal & Zyda, sanal ortamı, bilgisayar odaklı ve birden fazla katılımcılı büyük bir ağ olarak tanımlar (MMVE, *Massive Multiplayer Virtual Environment*) (Nevelsteen, 2018: 21; Singhal & Zyda, 1999). Brennan, ağ içindeki gerçekliğin sanal olabilmesi için mutlaka çevrimiçi (*online*) bir iletişimin kurulması gerektiğini belirtir (MMO, *Massive Multiplayer Online*) (Brennan, 2009). İşte bu noktadan itibaren çevrimiçi ağa katılanların amaçları doğrultusunda sürekli birbirleriyle etkileşim içinde olacak çoklu ortam fonksiyonları başlar. Nevelsteen, bu ortamın en dış kabuğunu doğrudan MMOW (*Massive Multiplayer Online World*) terimi ve türevleriyle açıklarken (Nevelsteen, 2018: 2); Gregory, böylesine sanal bir dünyanın kullanıcıları için ancak oyunla gerçeklik kazanabileceğini, bir başka ifadeyle ancak bilgisayar oyunlarının sanal dünyayı gerçek anlamda ortaya çıkarabileceğini vurgular (Gregory, 2018: 8). Bu bağlamda bugün yaygın olarak bilinen ve artık kavramlaşan "birden fazla katılımcılı devasa çevrimiçi rol yapma oyunları (MMORPG, *Massive Multiplayer Online Role-Playing Game*)", MMVE ile başlayan halkanın vazgeçilmez alanını oluşturur. Spence, sanal dünyanın halen tam olarak açıklanabilen bir terim olmadığını; aksine, her açıklamanın daima içinde uygulamadan uzak birtakım kavramlar bulundurduğunu belirtir (Spence, 2008: 5).



Şekil 1: Nevelsteen'e göre, sanal ortamı oluşturan katmanlar ve birbirleri arasındaki ilişkileri (Nevelsteen, 2018: 21)

Şekil 1'e göre Nevelsteen öncelikle, sanal ortamı sanal bir dünya olarak adlandırmanın yanlış olacağını vurgulamaktadır. Ona göre sanal ortam, kendi sanal dünyasını da içine alan en dış halka; bir başka ifadeyle, sanal bir evrendir. Sanal dünyaya ulaşmadan önce bu evreni önce çok sayıda kullanıcının oluşturması ve üstelik kullanıcıların çevrimiçi olması gerekir. Yani tek başına bir kullanıcı, ağda çevrimiçi olsa bile sanal ortamın bir parçası olamaz. Sanal dünya ancak bu noktadan sonra başlar. Bilgisayar oyunları ise bu dünyanın farklı bir boyutunu oluşturur. Oyunlarda gerçek anlamda bir sanal ortama ulaşılabilmesi için mutlaka oyuncuların kendilerine ait bir rol üstlenmeleri gerekir. Bu da her oyuncuyu sanal ortamda bir karakter sahibi yapar ve giderek ağ içinde birbirleriyle çevrimiçi etkileşime giren bir canlı karakterler bütünü ortaya çıkar.

MMORPG teriminden geriye dönüp bakıldığında, oyun kavramının sanal ortamı bugünlerin teknolojine getirdiği; bir başka ifadeyle, bilgisayar oyunlarının sanal ortamın geliştirilmesinde çok önemli faktör olduğu açıktır. Huizinga, kavramsal olarak oyunu tamamen fizyolojik bir olgudan veya fizyolojik olarak belirlenen psişik bir tepkiden daha fazla bir şey olarak tanımlar ve oyunu biyolojik veya en azından tamamen fiziksel bir faaliyetin sınırlarını aşan kültürden eski olabilecek bir kültür olarak görür (Huizinga, 1938: 17; Koçyiğit vd., 2007: 332). Huizinga'ya göre oyun, gerçekte vuku bulan fakat gerçek gibi görünüp gerçek olmayandır ve insanoğlu oyun oynayan, oyuncu olan ve oyuna meyledendir (Huizinga, 1938; Güven, 2022). Huizinga henüz 1938 yılında yalnızca oyun kavramını açıklamakla kalmamış aynı zamanda açık bir şekilde bugünün sanal ortamına neredeyse hükmeden bilgisayar oyunlarının bir tür tarifini yapmıştır. Üstelik çok önemli bir kavramı kullanarak: Gerçeklik (*reality*).

Çalcı, gerçekliği anonim bir süreç olarak tanımlar. Anonim ise isimsiz olan demekten çok eylemlerin ortaya çıkardığı, fiillerden kurulu bir gerçeklik imgesini ifade etmektedir (Çalcı, 2014: 199). Gerçeklik için kaynağı belli olsun veya olmasın mutlaka bir eylem gerekir. Saridakis ise bu tanımı bir adım daha ileriye götürerek gerçekliği modern fizikle malumat (*information*) arasında mutlaka olması gereken eylemsel bir adım olarak görür. Modern fiziğin müspet olgularına ulaşabilmek için en başında malumat olarak toplanan verilerin mutlaka bir gerçeklik sürecinden geçerek eyleme dönüşmesi gerekir (Saridakis, 2016: 329). Her iki açıklamayı birleştirecek olursak gerçeklik için, kaynağı belirli veya değil tüm hareketlerin/eylemlerin fiziksel ve müspet bir olgu yaratmasıdır diyebiliriz. Bu açıklama bilgisayar ortamına taşındığında, gerçekliği, doğrudan bir ya da uzak bağlantıyla birden fazla bilgisayarla bir araya gelen ve her biri bir rol üstlenmiş çok kişinin gerçekmiş hissiyle hareket ettiği/etkileşime girdikleri sanal bir ortam olarak düşünebiliriz. Günümüz bilgisayar teknolojisinde bu ortam kısaca Sanal Gerçeklik (SG, *Virtual Reality, VR*) olarak adlandırılmaktadır.

Bilgisayar oyuncuları son derece gerçekçi şekilde tasarlanmış etkileşimli üç boyutlu ortamları çok uzun zamandan bu yana deneyimlemektedirler. SG teknolojisi ise bu tecrübeyi daha yüksek bir seviyeye taşır. Oyunlarda temel unsur şans, bulmacalar, strateji, zamanlama gibi mekaniklerdir. SG oyunlar benzer mekaniklere sahip olabilecekleri gibi, oyun-dışı SG uygulamaları bu mekaniklerden

çok, hedeflenen tecrübenin kendisine veya o uygulamaya özel amaçlara odaklanır. Turizm, mühendislik, endüstriyel tasarım, mimarlık, emlâk, tıp, eğitim, eğlence vb. alanlarda oyun-dışı SG uygulamalarından faydalanılmaktadır (Linowes, 2015:6-8).

Her ne kadar teorik geçmişi perspektif ve film çalışmalarıyla 19.yy. sonlarına dek uzansa da bilinen ilk SG uygulamalarının Morton Heilig'in geliştirdiği simülatörüyle birlikte 1950'li yılların sonunda başladığını görmekteyiz. Heiling'in *sensorama* adını verdiği film seyretme makinesi, seyircinin bireysel olarak karşısına oturup gözlerini dayayarak yakınlaştığı bir mercekten neredeyse içindeymiş gibi hissettiği filmi seyretmesiyle bir tür simülatör özelliği taşımaktadır. Makinenin ancak 1980'li yıllarda yaygınlaşmasıyla birlikte konuyla ilgili çalışmalar artar. Özellikle Kanadalı yazar William Ford Gibson, *Neuromancer* romanında *cybernetic* ve *cyberspace* kavramlarıyla açıkladığı sanal gerçekliği öyle ifade etmiştir ki bilgisayar ve kullanıcısının sanal bir ortamda ama gerçekmiş hissiyle bugünkü gibi iletişime geçebileceği bir ortamı kurgulayabilmiştir. SG, katılımcılarına gerçekmiş hissi veren, bilgisayarlar tarafından yaratılan sanal ve dinamik bir ortamda karşılıklı iletişim olanaklarının sağlandığı bir benzetim modelidir (Bayraktar ve Kaleli, 2007: 2). Bir başka tanımlamada SG, bilgisayar ile oluşturulan çok boyutlu sanal ortamdaki şekillerin teknolojik donanımlar yardımıyla kişinin zihninde gerçek bir dünyada var olma hissi veren ve bu objelerle etkileşimde bulunmayı sağlayan bir teknolojidir (Şekerci, 2017: 1127). Sherman&Craig, SG'ı oluşturan dört önemli anahtar öğe sunar: Gerçeklik hissi verebilecek kadar sanal bir dünya (*virtual world*), sanal bir dünyada olduğunu hissettirecek kadar çok boyutlu bir çevre (*immersion*), olup biteni algıda hissettirecek kadar çevredekilerden geri dönüşler (*sensory feedback*) ve tüm bunları aynı ortamdaki diğerleriyle/her şeyle birlikte karşılıklı gerçekleştirebilme (*interactivity*) (Sherman&Craig, 2018: 6-15). Sherman&Craig'e göre ancak tüm bu ögeler birleştiğinde bir SG ortamından bahsedilebilir. Bu ve buna benzer SG tanımlarından yola çıkıldığında günümüz tüm SG üreticilerinin ortak noktası, bilgisayar ortamında kullanıcının gerçekmiş gibi hissedebileceği çok boyutlu sanal bir ortam yaratmak ve kullanıcıyı bu ortama sokabilmek için ona bazı yan donanımlar/kontrolcüler sunmaktır. Bir başka deyişle, bir kişinin sanal gerçeklik deneyimi yaşadığını söyleyebilmesi için, o kişinin kendisini çevreleyen ve fiziksel olarak orada olduğu hissini uyandıran yaratılmış bir dünya içerisinde bulunması, fiziksel eylemlerinin duyusal geri dönüş biçiminde karşılık bulması ve çevresiyle etkileşime geçebiliyor olması gerekmektedir. Bu anlamda bilgisayar oyun tasarımının vazgeçilmez araçları olan oyun motorları, sundukları olanaklar ve gelişmiş SG desteği sayesinde oyun-dışı SG uygulamaları için son derece uygun geliştirme ortamları olarak karşımıza çıkmaktadırlar (Keene, 2019). Sinclair, bir oyun motorunu grafik, ağ, fizik ve ses gibi özelliklerden sorumlu ve birbirleri ile etkileşim içerisinde bulunan alt sistemlerin bir arada bulunduğu bir yapı olarak tarif eder (Sinclair, 2020: 24). Geçmişten günümüze uzanan süreçte çeşitli platformlar (Windows, macOS, Linux, iOS, Xbox, PlayStation, Android vb.) için iki ve üç boyutlu oyun tasarlamaya olanak sağlayan çok sayıda oyun motoru ortaya çıkmıştır[1]. Bu

[1] https://en.wikipedia.org/wiki/List_of_game_engines

araçlardan bir tanesi olan *Unity*[2], iki veya üç boyutlu uygulamaların birden fazla platform için derlenebildiği çok platformlu (*cross-platform*) bir oyun motorudur. *Unity*, basit arayüzü ve geniş geliştirici kitlesi sayesinde SG topluluğunun tercih ettiği bir oyun motoru haline gelmiştir (Keene, 2019).

Şimdilerdeyse SG temel özelliklerini kapsayan ancak ayrıcalıklı bazı özellikleriyle kendine ait bir sanal ortam haline gelen popüler bir sanal ortam/kavram daha mevcut: *Metaverse*. Facebook yaratıcısı Zuckerberg'in *Meta* adıyla kurduğu yeni şirketiyle daha da popüler hale gelen *metaverse*, sanal ortamın tüm yönleriyle şimdiye kadarki geliştirilmiş en etkili gerçekliktir. En ilk Snow Crash bilimkurgu romanında kelimeyi kullanan yazar Stephenson, *metaverse* ortamını yaşayan bir sanal akıcı hayat olarak kurgulamıştır (Stephenson, 1992). Dolayısıyla ortamın kendine özgü ve kararlı bazı kuralları vardır: Akıcı bir hayat, yani *metaverse* ortamdan ayrılınsa da tekrar girildiğinde ayrı kalınan zamanın *metaverse*'de devam ediyor olması. Avatar, yani her kullanıcının kendini *metaverse*'de temsil edeceği vücut bulan sanal kişiliği. Maddiyat, yani tıpkı gerçek hayatta olduğu gibi hayatın *metaverse* ortamda sürdürülebilmesi için gerekli ekonomik varlık. Bunlar dışında kalan çok kullanıcılı olma, kullanıcının *metaverse*'deki hareketlerinin aynı anda gerçek hayatta da yapması (senkron) vs. gibi aslında SG'devar olan özellikler *metaverse*'te de mevcuttur. Şimdiye kadar Stephenson'un kurguladığı ortama yakın herhangi bir uygulama henüz üretilebilmiş değil. Ancak sanal para ve borsası gibi bazı *metaverse* özellikleri günümüzde aktif hale gelmiştir. Bununla birlikte avatar ve sanal ortam yaşamı (araziler, arsalar, kentler, merkezler vs.) gibi bazı diğer özellikler ise henüz emekleme aşamasındadır diyebiliriz. Spielberg filmi 2018 yapımı "Ready Player One" ise, *metaverse* ortamını en iyi anlatan filmler arasında ilk sıralarda gösterilebilir.

Sanal Gerçeklikte Ses, Müzik ve Performans

Sanal dünyada işitsel bir sunum en az görsel kadar önemlidir. Ses, kullanıcıların sanal dünyada içerisindeki zihinsel varoluş algısına önemli katkı sağlar. Boyut, doğa ve ortam atmosferi hakkında ipuçları veren ortam (*ambient*) seslerinden, karakterler ve belirli nesneler ile ilişkilendirilmiş seslere kadar sanal ortamdaki tüm sesler, kullanıcı tarafında kavrayış ve keyif alma boyutunda önemli rol oynar (Sherman & Craig, 2018: 225).

Günümüz sanal ortam teknolojileri eğitimden sağlığa, ticaretten seyahate birçok alanda kullanılır olduğundan insan hayatını artık doğrudan etkiler hale gelmiştir. Özellikle tüm dünyayı etkisi altına alan Covid-19 salgını, sanal ortam teknolojilerini çok daha önemli bir konuma taşımıştır. Şüphesiz her alanda etkisini göstermeye devam eden böylesi bir teknolojinin müzikle buluşması da kaçınılmazdı. Bilgisayar oyunlarıyla birlikte gelişimini sürdüren ve özelliği gereği sanal ortama en ilk uyum sağlayan alanlardan biridir müzik. Ancak mesele sanal bir ortamda performansa geldiğinde işler

[2] https://unity.com

henüz başlangıç seviyesindedir. Dolayısıyla, *metaverse* için gelecekte gerçek örneklerine rastlanacak olan sanal ortamda müzik performansı, şimdilerde daha çok SG teknolojisiyle iç içe bir seyir izlemektedir. Bunda elbette bilgisayar oyunlarının ve oyun şirketlerinin payı büyüktür. Örneğin popüler bilgisayar oyunlarından Fortnite içindeki 2019 tarihli Marshmallo konseri, özelde *metaverse* genelde/gerçekte SG konserlerinin ilki olarak kabul edilmektedir (Güven, 2022). Bu tür büyük organizasyonlarda asıl amaç, konseri ve konser ortamını sanal gerçekliğe taşımaktır. Müzisyen(ler) bir mekânda gözlükleri ve olası giyilebilir diğer sanal ortam donanımlarını/kontrolcüleri takarak sahne performansı yaparken, dünyanın dört bir yanındaki hayranları da sanal gözlüklerini takarak gerçek konser yerindeymiş gibi konsere katılabilir. Böyle bir sanal gerçeklik etkinliği aslında hayatın olağan akışında, yani akıcı bir hayatta gerçekleşir. Katılımcı, konserden (sanal ortamdan) ayrılsa da konser devam etmektedir ve katılımcı geri geldiğinde sanal ortamdaki zaman akmaya devam eder. Diğer taraftan tüm katılımcıları ortamda avatarı temsil edebildiği gibi konserdeki müziklerin ve organizasyonun telif hakları da korunabilir, yani etkinliğin bir ekonomisi vardır. Tüm bunlar göz önüne alınırsa, aslında müzik üretimine yönelik sanal ortam uygulamalarının gelecekte daha çok *metaverse* ile uyumlu olabileceğini söyleyebiliriz. Örneğin geçtiğimiz yıl Epic Games'in Fortnite oyununda gerçekleştirilen Travis Scott ve Ariana Grande konserleri, telif haklarından promosyon amaçlı malzeme satışı ve turne programına kadar sanal ortam ekonomileriyle ön plana çıkmıştır (Güven, 2022).

Müzikte sanal gerçeklik konserlerinin özellikle son 1-2 yılda giderek arttığını görmekteyiz. 2020 sonrasında Kanadalı şarkıcı The Weeknd'in Blinded by the Lights şarkısını ve Forbes şirketinin dünyayı sarsan oyunu Pokemon'un 25. yılı kutlamalarında Amerikalı rap yıldızı Post Malone (Austin Richard Post) konserini *metaverse* üzerinden dinleyicilerle buluşturması, sanal gerçeklik ortamındaki müziğin oyun şirketlerinin tekelinden giderek çıkabileceğin ilk sinyallerini vermiştir. Öyle ki Justin Bieber, sanal eğlence şirketi Wave aracılığıyla ilk *metaverse* konserini geçtiğimiz yıl gerçekleştirmiştir. Bu gelişmelere kayıtsız kalamayan dünya devi müzik prodüksiyon şirketlerinden Warner Music Group ise Justin Bieber, Cardi B. ve Ariana Grande ile sanal gerçeklik konser anlaşmalarını hiç vakit kaybetmeden yapmıştır bile. Gelişmeleri yakından takip eden Universal Music Group; Billie Eilish ve Taylor Swift gibi yıldızlarını *metaverse*'e taşımak için dijital avatar şirketi "Genies" ile ortaklık kurmuştur. Diğer taraftan, kullanıcıların kendi oyunlarını programlamalarına ve oluşturmalarına olanak tanıyan popüler bir çevrimiçi oyun olan Roblox, Kai sanal karakteri aracılığıyla uzun soluklu sanal gerçeklik konserleri düzenlenmektedir[3]. Bu gelişmelerin ardından bir diğer dünya devi prodüksiyon şirketi Sony Music, Roblox ile sanal gerçeklik konserleri üzerinde iş birliği yapacağını duyurmuştur. Müzik grubu *Muse* vs. örneklerde olduğu gibi son yıllarda *metaverse*

[3] Son yıllarda artan sanal gerçeklik konserleri hakkında daha detaylı bilgi için Bkz.: http://t.ly/5y9n

ve SG konserlerin sayısı giderek çoğalmakla birlikte, müzisyen veya dinleyicilere interaktif konser deneyimi yaşatmayı vadeden NoysSG[4] gibi şirketlerin sayısı da giderek artmaktadır.

Bir müzik performansının sanal gerçeklik üzerinden dinleyiciyle buluşturulması her ne kadar kulağa hoş gelse de teknik olarak görsel-işitsel bir akışın (*streaming*) ağ üzerindeki hareketi aslında sanal gerçeklikten çok daha önceleri üzerinde çalışılan bir konu olarak karşımıza çıkar. Müzik tarafındaki bilinen adıyla Ağ Üzerinden Müzik Performans (*Networked Music Performance*, NMP), en ilk Amerikalı bağdar Pauline Oliveros tarafından 1991 yılında telefon hattı üzerinden görüntülü müzik iletimi olarak denenmiştir. Günümüzdeki gibi bir teknolojik alt yapının o yıllarda olmadığı düşünülürse, böylesi bir denemenin sonuçlarını da tahmin etmek güç değil. Ancak, 2008 yılında Stanford Pan-Asian Festivali kapsamında Pekin Üniversitesi ve Stanford Üniversitesi arasında "Pasific Rim of Wire" adlı orkestral bir yaratının internet üzerinde seslendirimi, günümüz sanal gerçeklik üzerinde denenen müzik performanslarının başlangıcını oluşturur (Cáceres vd., 2008). Tahmin edilebileceği gibi NMP uygulamalarında ortaya çıkan en büyük sorun zamansal gecikme *latency*[5] olmasına rağmen, 21.yy. sonrası gelişen teknolojiyle birlikte artan internet hızları ve gelişen alt yapı, *latency* sorunlarını yok denecek kadar az seviyelere indirmiştir veya indirmesi beklenmektedir.

Sanal Gerçeklikte Etkileşimli Ses-Müzik Tasarımı

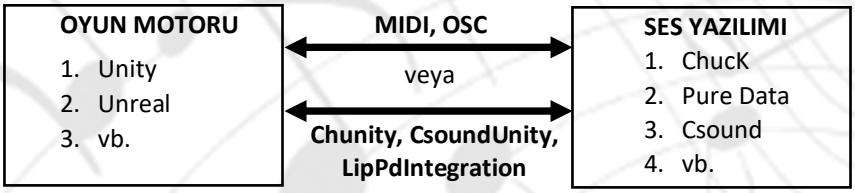
Her ne kadar *metaverse* için oyun devleri çoktan harekete geçmiş olsa da sanal bir ortamda herkesi tatmin edebilecek tam bir müzik performansı günümüz teknolojisinde henüz çok erken bir beklenti. Üstelik buna bir de canlı çalgı performansları eklendiğinde beklenti süresini birkaç yıl daha ötelemek gerek. Peki müzik için SG çok mu gerekli? Müzikte eğitim, araştırma, teori vs. için evet, ancak konser gibi performansa dayalı etkinlikler için şüphelerimiz yok değil. Bu çalışmada, bir sanal gerçeklik ortamı içerisinde yer alan nesneler (kullanıcının kendisi de dahil olmak üzere) arasındaki etkileşimin ve bu etkileşimlerden elde edilecek verinin, elektronik müzik performansı bağlamında yordamsal ses tasarımı içerisinde ne şekilde değerlendirilebileceğine ilişkin yöntemler değerlendirilmiş olup, bu yöntemlerden yararlanılarak, sanal bir mekan içerisinde kullanıcının bir SG gözlüğü ve kontrolcüler aracılığıyla etkileşime geçerek ses üretebildiği iki elektronik çalgı örneği tasarlanmış ve sunulmuştur. Yordamsal ses kullanımı, tasarımın öne çıkan unsurlarından biridir.

Geleneksel ses teknolojilerinin temelini oluşturan ses kayıt sürecinde, gerçek dünya sesleri bir mikrofon aracılığıyla yakalanır, mikslenir, işlenir ve *mastering* sürecinin ardından son şeklini alır. Bu şekilde oluşturulmuş ses efektleri ve müzikler değişmez yapıdadır ve her seslendirildiklerinde aynı kalırlar. Buna karşılık dinamik bir ses tasarımına olanak sağlayan yordamsal ses, bilgisayar temelli ses ve müzik tasarımına dayalı, doğrusal olmayan (*non-linear*), sentetik ve canlı bir girdiyle birlikte

[4] https://www.noysvr.com/
[5] Müzik performanslarında *latency* sorunu üzerine yapılan çalışmaların süreci ve ayrıntıları için Bkz.: Davies G. (2015), "The Effectiveness of LOLA (LOw LAtency) Audiovisual Streaming Technology for Distributed Music Practice", MA thesis, Edinburgh Napier University.

programlamaya dayalı kurallara bağlı gerçek zamanlı bir ses üretme biçimi olarak karşımıza çıkar (Farnell, 2007: 1-2). Bir başka deyişle yordamsal ses, parametreleri kullanıcının etkileşimi veya ortamın özelliklerine bağlı olarak değişebilen algoritmalar sayesinde dinamik ses-müzik tasarımına olanak sağlar.

Yordamsal ses tasarımı, bir oyun motorunun doğrudan kendi ortamında[6] veya ses programlama amaçlı tasarlanmış bir veya birden çok programlama dili ile gerçek zamanlı iletişim kurulması biçiminde gerçekleşebilir. Birinci yöntemin, oyun motorunun bu amaca yönelik olası sınırlılıkları ve belirli bir seviyede ses programlama bilgisi gerektirmesi açısından olumsuz yönlere sahip olacağı söylenebilir. İkinci yöntemde, oyun motorunun arka planda çalışan bir ses yazılımıyla OSC[7] veya MIDI[8] gibi bir protokol aracılığıyla gerçek zamanlı iletişime geçerek karşılıklı parametre aktarımında bulunması gerekir. Bu yöntemde, ses üretim süreci ile ilgili tüm sorumluk ses yazılımında olacağından yazılımcının üstlenmesi gereken programlama yükü büyük oranda azalacaktır. Buna karşılık, MIDI kullanımında yazılımlar arası veri aktarımı için ihtiyaç duyulacak üçüncü parti sanal MIDI kabloları[9] gibi aracı yazılımlar ve MIDI veri yapısının sınırlılıkları ile OSC kullanımı sırasında karşılaşılabilecek olası ağ kaynaklı aksaklıklar bu yöntemin olumsuz yönleri olarak öngörülebilir.



Şekil 2: Popüler oyun motorları, kodlamaya dayalı ses yazılımları ve her ikisi arasında veri iletim araçları (protokoller veya eklentiler).

Ses yazılımının oyun motorunun bir bileşeni haline getirilmesi, böylelikle iki ortamın bir eklentiyle haberleşmesinin sağlanması, sözü geçen olumsuzlukları ortadan kaldırmak adına son derece uygun bir çözüm olacaktır. Bu amaçla *Chunity*, CsoundUnity[10] ve LibPdIntegration[11] eklentileri sırasıyla, gerçek zamanlı ses tasarım/programlama ortamları olan *ChucK*[12], Csound[13] ve Pure Data[14] ile *Unity* oyun motoru arasında entegrasyonu sağlamak amacıyla geliştirilmiş yazılımlar/eklentilerdir.

[6] MCV Staff, https://www.mcvuk.com/development-news/procedural-audio-with-unity
[7] Wright, M., Freed, A., & Momeni, A. (2003). Open Sound Control: State of the Art. *A NIME Reader*, 125-145
[8] MIDI Association, https://midi.org
[9] Tobias Erchsen tarafından Windows işletim sistemi için geliştirilmiş olan "loopMIDI" bu yazılımlardan bir tanesidir (https://www.tobias-erichsen.de/software/loopmidi.html). Apple'ın macOS işletim sistemi içerisinde yer alan "IAC Driver" bu desteği dahili olarak vermektedir. (https://support.apple.com/guide/audio-midi-setup/transfer-midi-information-between-apps-ams1013/mac).
[10] Walsh, R. (2015). *Csound and Unity3D*. In ICSC.
[11] Moody, N. (2018). LibPd Unity Integration: A Libpd Wrapper for Unity. https://github.com/LibPdIntegration/LibPdIntegration
[12] Wang, G., & Cook, P. R. (2003). *ChucK: A Concurrent, On-The-Fly, Audio Programming Language*. In *ICMC*.
[13] Lazzarini, V., Yi, S., Heintz, J., Brandtsegg, Ø., & McCurdy, I. (2016). *Csound: A Sound and Music Computing System*. Springer.
[14] Puckette, M. S. (1997). *Pure Data*. In *ICMC*.

Chunity:

Chunity[15], kendine özgü zamansal bir tasarımı, kesin ve anlatımsal eş zamanlı bir programlama modeline sahip ve kodun anında eklenip değiştirilebilmesine olanak sağlayan ses sentezleme/analiz amaçlı bir programlama dili olan *ChucK*[16] ile *Unity* ortamında etkileşimli görsel-işitsel uygulamalar geliştirmeye imkân sağlayan bir yazılım/eklentidir.

ChucK programlama dilinde basit bir sinüs dalga üretecine ait kod örneği aşağıda gösterilmektedir. Birinci satırda *SinOsc* birim üreteci (*unit generator*) tipinde *sine* isimli bir sinüs dalga üreteci tanımlanmış ve *dac* ile gösterilen ses çıkışına "=>" operatörü ile bağlanmıştır. Bu operatöre *ChucK* operatörü adı verilir. Soldan sağa doğru bir ok biçimindeki bu operatör, bir sinyalin bir taraftan diğer tarafa doğru akış yönünü temsil etmek amacıyla tasarlanmıştır. İkinci satırda sinüs üretecinin genlik değeri 0.5, üçüncü satırda ise üretecin frekansı 440Hz olarak belirlenmiştir. Son satırdaki ifade ile kodun çalıştırıldığı andan itibaren (*now*) 1 sn boyunca sinyal üretimi sağlanmıştır (Kapur vd. 2015: 19). Aşağıda (a) örneğinde, üretecin genlik ve frekans değerleri sabittir. *ChucK* ortamında da pek çok programlama dilinde olduğu gibi, değişkenler aracılığıyla, performans sırasında çeşitli parametrelerin dinamik olarak değiştirilebilmesi mümkün hale gelir; (b) örneğindeyse, aynı üretecin frekans değeri bir *f* değişkeni aracılığıyla atanmıştır.

(a)
```
//Sabit frekanslı sinüs osilatoru
SinOsc sine => dac;
0.5 => sine.gain;
440 => sine.freq;
1::second => now;
```

(b)
```
//Sabit frekanslı sinüs osilatoru
float f;
440.0 => f;
SinOsc sine => dac;
0.5 => sine.gain;
f => sine.freq;
1::second => now;
```

Chunity eklentisi, *Unity* paket yöneticisi (*package manager*) içerisinden ilgili projeye dahil edildikten (*import*) sonra sahne (*scene*) içerisinde boş bir nesne oluşturularak bu nesneye bir *ChuckMainInstance* bileşeni eklenir. Bir nesne üzerinde *ChucK* kodu çalıştırabilmek için nesnenin *ChuckMainInstance* nesnesine referans içeren ayrıca bir *ChuckSubInstance* bileşenine de sahip olması gerekir. Böylelikle bu nesnenin sahip olduğu tüm betikler içerisinde *ChucK* eklentisine ulaşmak ve kod çalıştırmak mümkün hale gelir. *Unity* içerisinde *ChucK* kodu, *ChuckSubInstance* bileşeninin *RunCode* fonksiyonu ile çalıştırılır.[17]

Aşağıda, *ChucK* kodunun bir nesneye ait *Start()* fonksiyonu içerisinde çalıştırıldığı bir örnek görülmektedir. Örnekte, frekansı 50 milisaniyede bir, 1000Hz ile 2000Hz aralığında değişen bir sinüs sinyali üreten kod, nesne başladığı anda çalıştırılır ve nesne sahnede kaldığı sürece çalışmaya devam eder.

[15] Atherton, J., & Wang, G. (2018). *Chunity: Integrated Audiovisual Programming in Unity*. In *NIME*. pp. 102-107.
[16] ChucK: Strongly-Timed, Concurrent, and On-The-Fly Music Programming Language. 2022 August, 2, https://chuck.stanford.edu
[17] *Chunity: Tutorials.* 2022, August, 3. https://chuck.stanford.edu/chunity/tutorials

```
void Start ()
{
   ChuckSynth = GetComponent<ChuckSubInstance> ();
   ChuckSynth.RunCode(@"
      SinOsc sine => dac;
      While(true)
      {
         Math.random2f(1000i 2000) => sine.freq;
         50:ms => now;
      }
      ");
}
```

Chunity, genel (*global*) değişkenlerin kullanımıyla *ChucK* ile *Unity* arasında veri aktarımına da olanak sağlar. Aşağıdaki örnekte, bir nesnenin Y eksenindeki konumu "a" genel değişkeni aracılığıyla, her bir karede çalıştırılan *Update()* fonksiyonu içerisinde *ChucK* koduna aktarılmakta ve osilatörün genlik değerini değiştirmektedir. Benzer şekilde *GetFloat* fonksiyonu ile *ChucK* kodundan *Unity*'e veri aktarımı da mümkündür.[17]

```
void Start()
{
   ChuckSynth = GetComponent<ChuckSubInstance>();
   ChuckSynth.RunCode(@"
      global float a;
      SawOsc saw => dac;
      440 => saw.freq;

      while(true)
      {
         a => saw.gain;
         ms => now;
      }
   ");
}

void Update()
{
   ChuckSynth.SetFloat("a", transform.position.y);
}
```

Yordamsal Ses-Nesne Etkileşimi

Sanal bir ortamda tıpkı gerçek dünyada olduğu gibi ortamı meydana getiren her şey bir nesnedir ve nesneler arasındaki ilişkiler etkileşimi meydana getirir. Sanal dünyaların tasarımına olanak sağlayan araçlar olan oyun motorlarında da tüm nesneler oyun nesnesi olarak adlandırılırlar. Gregory, bir oyun dünyasında yer alan nesneleri statik ve dinamik olmak üzere iki kategoriye ayırır. Arazi, bina, yol, köprü gibi oyunla etkileşime girmeyen, yani hareketsiz olanlar statik; karakterler, taşıtlar, silahlar, dinamik aydınlatmalar vs. olanlar dinamik nesnelerdir (Gregory, 2018: 56). *Unity* ortamında statik veya dinamik fark etmeksizin, bir oyun dünyasında yer alan tüm nesneler oyun nesnesi olarak

tanımlanır. En temel nesneler olan oyun nesneleri tek başlarına etkin olmamakla birlikte, kendilerine işlevsellik kazandıran bileşenler (*components*) için birer taşıyıcı (*container*) görevi üstlenirler[18]. Goldstone, *Unity*'nin de bünyesinde yer alan üç boyutlu mekân (sanal ortam) tasarımının önemli unsurlarını nesnelere göre şu şekilde sınıflandırmakta ve açıklamaktadır (Goldstone, 2009: 9-13):

Koordinatlar: Nesnelerin üç boyutlu dünyadaki konum, boyut ve dönüş değerlerinin ifade edilebileceği üç eksenli (X, Y, Z) bir kartezyen koordinat sistemidir. Bu sistemde merkez (0, 0, 0) noktasıdır.

Vektörler: Üç boyutlu ortamda yön belirten niceliklerdir. Nesnelerin yönleri ile aralarındaki göreceli açıların ve uzaklıkların hesaplanabilmesi için önemli bileşenlerdir.

Kameralar: Oyun nesneleri ve karakterler ile ilişkilendirilebilen veya üç boyutlu dünyanın herhangi bir noktasında yerleştirilebilen kameralar temel olarak bu dünyanın görüş alanını belirlerler.

Poligonlar, Kenarlar ve Köşeler: Üç boyutlu dünyada nesneler birbirine bağlanmış çok sayıda iki boyutlu poligondan meydana gelir. Kenar bir poligonun bir kenarı, köşeler ise bu kenarların buluştuğu noktalardır. Birbirleri ile ilişkili çok sayıda poligon bir araya gelerek *mesh* olarak bilinen karmaşık şekilleri oluştururlar.

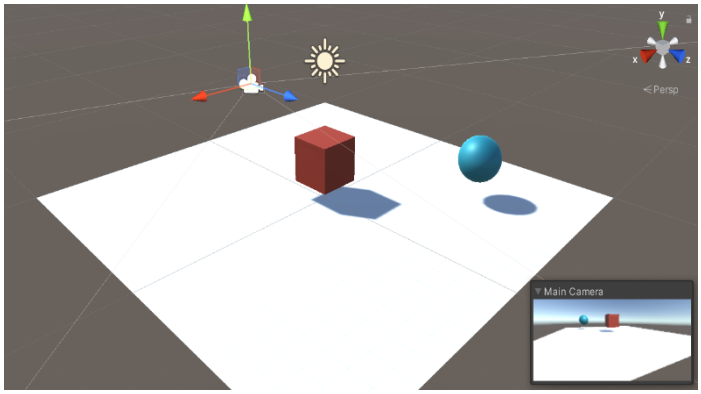
Materyaller, Dokular ve Gölgelendiriciler: Materyaller üç boyutlu nesnelerin görünüşünü belirler. Bir materyal sadece basit temel renklerden oluşabileceği gibi, doku olarak adlandırlan görüntülerden de meydana gelebilir. Materyaller, nesnelerin yansıtıcılık gibi görsel davranışlarını belirleyen betikler olan gölgelendiriciler ile çalışırlar.

Katı Gövdeler: Oyun motorlarında nesnelere gerçek dünyadaki fiziksel özelliklerini ve davranışlarını kazandırmak amacıyla verilen katı gövde özelliği, eklendiği nesnenin kütle, yer çekimi, hız ve sürtünme gibi özelliklere sahip olmasını sağlar.

Çarpışma Algılama: Üç boyutlu dünyada yer alan nesnelerin birbirleriyle teması, bir başka deyişle çarpışmaların algılanması, nesnelere eklenen çarpıştırıcı (*collider*) bileşeni sayesinde gerçekleşir. Çarpıştırıcı, algıladığı çarpışmayı fizik motoruna bildirir. Fizik motoru çarpışmanın yönü, hızı ve diğer faktörlere bağlı olarak nesnenin ne şekilde tepki vereceğini belirler.

Özetle, sanal bir dünyada poligonlar ve materyaller ile şekillendirilmiş nesneler üç boyutlu bir koordinat sistemi içerisinde yer alırlar. Katı gövde özelliği, eklendiği nesnelere gerçek dünyadakine benzer fiziksel davranışlar kazandırırken, bu nesneler arasındaki etkileşim yine aynı fizik kurallarına göre gerçekleşir.

[18] *Unity Documentation: Unity Manual.* 2022 August, 6. https://docs.unity3d.com/Manual/UnityManual.html



Şekil 3: *Unity* koordinat sistemi içerisinde bir yüzey (*plane*), küp, küre, yönsel ışık kaynağı ve kamera. Kameranın görüş alanı sağ alttaki pencerede yer almaktadır.

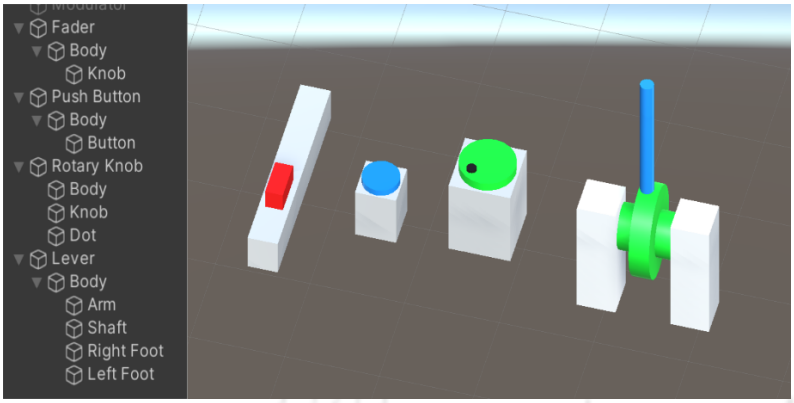
Dinamik ve hareketli bir dünya içerisinde yer alan tüm nesneler kaçınılmaz olarak birbirleriyle etkileşim içerisinde bulunurlar. Etkileşim denildiğinde akla ilk gelen iki veya daha fazla nesnenin birbirleri ile temas düzeyindeki ilişkisi olsada, aynı ortam içerisinde bulunan tüm nesneler konum ve mesafe gibi özellikleri açısından sürekli ve dinamik bir ilişki içerisindedirler. Bir oyun motoru içerisinde yaratılmış sanal bir ortamda nesneler arasındaki bu ilişkilerin sayısal olarak ifade edilebilmesi, etkileşimli bir ses/müzik tasarımı sürecinde değişkenler olarak kullanılabilmelerine olanak sağlar. *Unity*'de tüm oyun nesneleri, nesnenin konum (*position*), dönüş (*rotation*) ve ölçek (*scale*) özelliklerini belirleyen bir dönüşüm (transform) bileşenine sahiptir. Çarpıştırıcı bileşeninin de devreye girmesiyle, nesnelerin konumları, aralarındaki mesafe ve nesneler arasındaki çarpışmalardan elde edilecek veri ve tetiklenen eylemlerin yordamsal ses tasarımı sürecinde ne şekilde kullanılabileceğine yönelik yöntemler örnekleriyle aşağıda sunulmuştur.

Konum:

Konum, bir nesnenin üç boyutlu koordinat sistemi içerisinde X, Y ve Z eksenlerine göre bulunduğu yeri belirtir. Dönüş, bir nesnenin X, Y ve Z eksenleri etrafındaki derece cinsinden açısıdır. Ölçek ise bir nesnenin orijinal boyutuna göre herhangi bir eksendeki (veya tüm eksenlerde orantılı olarak) oranıdır. *Unity* ortamında önemli kavramlardan bir tanesi de ebeveynlik (*parenting*) ilişkisidir. Nesneler, hiyerarşik bir yapı içesinde birbirlerine göre alt (*child*) veya üst (*parent*) nesne konumunda bulunabilirler. Alt konumda bulunan bir nesnenin *transform* değerleri, bağlı olduğu üst nesneye göre bağıl olarak görünür. Bu değerlere yerel koordinatlar (*local coordinates*) adı verilir. Alt nesne konumunda bulunmayan nesnelerin oyun dünyasındaki transform değerleri ise genel koordinatlar (*global coordinates*) olarak adlandırılır[19]. Şekil 3'de yer alan tüm nesneler aralarında bir ebeveynlik ilişkisi bulunmayan nesnelerdir ve konumları genel koordinat sistemine göre belirlenmiştir. *Unity*'de bir oyun dünyasının büyüklüğü göz önünde bulundurulduğunda, bir nesnenin genel koordinatının

[19] *Unity Documentation: Unity Manual.* 2022 August, 6. https://docs.unity3d.com/Manual/UnityManual.html

müzikal bir parametreye yönlendirilmek için pek de kullanışlı bir veri olamayacağı açıktır. Buna karşılık, ebeveynlik ilişkisi içerisinde tasarlanmış, daha sınırlı bir aralıkta değişen değerler üretebilen sanal nesneler, müzikal parametrelerin kontrolü için daha uygun araçlar haline gelebilirler.



Şekil 4: *Unity*'de ebeveynlik ilişkisi içerisinde tasarlanmış *fader*, buton, döner buton ve bir manivela (sağda) ve bu araçları oluşturan nesnelerin hiyerarşik yapısı.

Şekil 4'te çeşitli amaçlar için kullanılabilecek *fader*, buton, döner buton ve bir manivela tasarımı ile bu araçları meydana getiren nesnelerin hiyerarşik yapısı görülmektedir. Bu araçlardan *fader* nesnesi ile müzikal bir parametrenin ne şekilde kontrol edilebileceğine ilişkin örnek kod aşağıda sunulmuştur:

```
void Update()
{
   knobPosition = faderKnob.transform.localPosition;
   knobPosition.z = Mathf.Clamp(faderKnob.transform.localPosition.z, -0.2f, 0.2f);
   faderKnob.transform.localPosition = knobPosition;
   value = (knobPosition.z + 0.2f) * 2.5f;
   sineSynth.SetFloat("g", value);
}
```

Yukarıdaki örnekte, *fader* nesnesinin hareketli bölümünün (kırmızı kısım) gövdesine (*body*) göre Z eksenindeki konumu (*localPosition*) 0–1 aralığına normalleştirildikten sonra *ChucK* kodu içerisinde bulunan sinüs osilatörünün genliğini kontrol edecek şekilde yönlendirilmiştir. Benzer şekilde, döner buton ve manivela gibi nesnelerin hareketli kısımlarının dönüş açı değerleri de uygun bir aralığa dönüştürülerek herhangi bir müzikal parametrenin kontrol edilmesi de sağlanabilir.

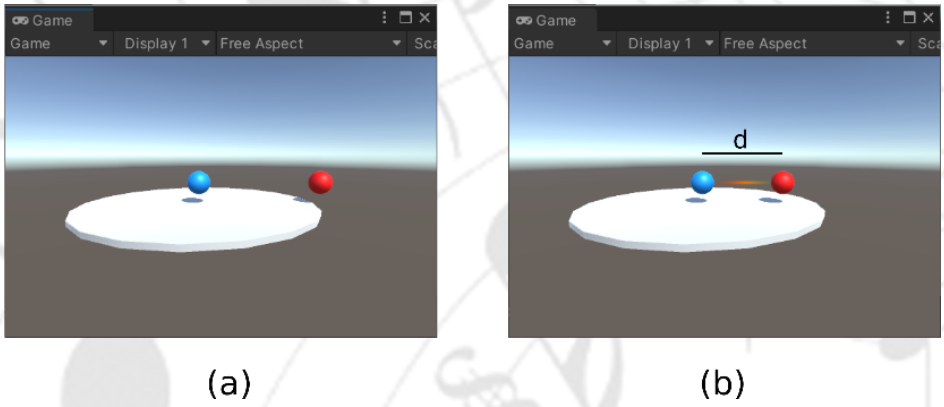
Mesafe:

Nesneler arasındaki mesafe değişiminin müzikal parametrelerin kontrol edilmesi prensibine dayalı çalgı tasarımlarını gerçek hayatta da görmek mümkün. Theremin belki de buna en iyi örnek olarak verilebilir. Sol ve sağ tarafında antene benzer çıkıntılarıyla bir kutu biçimindeki bu çalgı, müzisyenin elleri ile antenler arasındaki mesafeyi değiştirerek sesin frekansını (sağ el) ve genliğini (sol el) kontrol edebilmesine olanak sağlar. Çalgının, Maku[20] tarafından geliştirilmiş bir SG

[20] https://www.oculus.com/experiences/quest/3497148190407250

uygulaması da mevcuttur. *Reactable*[21] (Kaltenbrunner, 2006) ise her biri farklı bir işleve sahip objelerin, görsel tepki verebilen aydınlatmalı bir masa üzerine yerleştirilerek, aralarındaki mesafelere bağlı olarak birbirleri ile etkileşimli biçimde ses üretmesi prensibine dayanmaktadır.

Sanal bir dünyadaki koordinat sisteminde farklı konumlarda bulunan nesneler arasındaki mesafeler de uygun bir biçimde düzenlenerek müzikal parametrelerin kontrolünde kullanılabilir. Ancak, tıpkı genel koordinatların kullanılmasında olduğu gibi, büyük bir dünyada birbirlerinden çok uzakta bulunan iki nesne veya kullanıcı ile görüş alanı içerisinde bulunmayan bir nesne arasındaki mesafe gibi göreceli büyük değerler bu amaca yönelik uygun veri niteliği taşımayabilir. Bu bağlamda, iki nesne arasındaki mesafeye dayalı etkileşimin bir uzaklık eşiği ile sınırlandırılması, gerek kullanıcı etkileşim alanının sınırlarının belirlenmesi, gerekse bu alanın dışında kalan ses kaynaklarının devre dışı bırakılabilmesine bağlı olarak performansın iyileştirilebilmesi gibi avantajlar sunacaktır. Bu yöntemde, iki nesne arasında ölçülen uzaklık, belirli bir uzaklık eşiğinin üzerinde olduğu sürece dikkate alınmaz. Mesafe eşik değerinin altına indiğinde, ölçülen uzaklık değeri uygun bir aralığa dönüştürülerek müzikal bir parametreye yönlendirilir.



Şekil 5: İki nesne arasında belirlenmiş uzaklık eşiği

Şekil 5'teki örnekte mavi küre, sinüs dalga üreteci kodu içeren bir *ChuckSubInstance* bileşenine sahiptir. Kırmızı küre, beyaz daire ile belirlenmiş alanın dışında olduğu sürece *ChuckSubInstance* bileşeni devre dışı durumdadır. Küre alanın içerisinde girdiği anda *ChuckSubnstance* etkin hale geçmekte ve iki nesne arasındaki uzaklık ChucK koduna modülasyon parametresi olarak aktarılarak nesnelerin yakınlığına bağlı olarak vibrato derinliğinin kontrol edilmesi sağlanmaktadır. Uygulamaya ait *update()* fonksiyonu aşağıda görülmektedir:

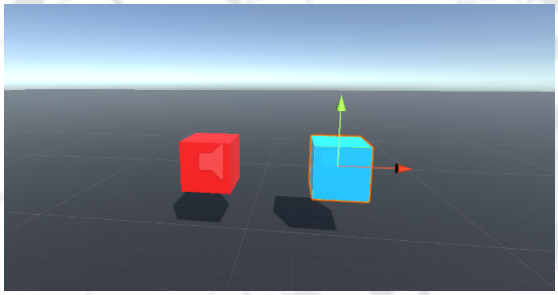
```
void Update()
{
   Vector3 carrierPos = thisCarrier.transform.position;
   Vector3 modulatorPos = modulator.transform.position;
   dist = Vector3.Distance(carrierPos, modulatorPos);
```

[21] https://www.upf.edu/web/sergi-jorda/reactable

```
    if (dist < 5)
    {
        ChuckSyn.enabled = true;
        ChuckSyn.SetFloat("modulation", (5 - dist) * 10);
    }
    else
    {
        Line.enabled = false;
        ChuckSyn.enabled = false;
    }
}
```

Çarpışma:

Unity'de iki nesne arasındaki çarpışmanın algılanması, *collider* bileşeni ile mümkündür. Çarpışma algılandığı anda *OnCollisionEnter*, çarpışmanın devam ettiği süre boyunca *OnCollisisonStay*, iki nesne arasındaki temasın sona ermesi ile *OnCollisionExit* fonksiyonları çalıştırılır[22]. Çarpışma, oyun dünyasında yer alan iki nesne arasında gerçekleşebileceği gibi, oyuncu ile herhangi bir nesne arasında da gerçekleşebilir. İkinci senaryo, özellikle sanal gerçeklik araçlarının kullanıldığı durumlarda, kullanıcının elleri ile ortamdaki nesneler arasındaki etkileşimin müzikal parametrelere yönlendirilebilmesine olanak sağlar.



Şekil 6: Bir ChuckSubInstance ve cubeController betiği içeren küp (kırmızı) ile çarpışmanın gerçekleştirilebilmesi amacıyla yerleştirilmiş küp (mavi).

Şekil 6'da bir oyun sahnesi içerisine yerleştirilmiş iki küp görülmektedir. Kırmızı kübe bir *ChuckSubInstance* bileşeni ve çarpışmanın denetiminin kontrol edildiği bir *CubeController* betiği eklenmiştir. Betik içerisindeki *ChucK*[23] ile *OnCollisionEnter()* ve *OnCollisionExit()* fonksiyonlarına ait kod parçaları aşağıda sunulmuştur.

```
private void OnCollisionEnter(Collision collision)
{
    if (lisTriggered)
    {
        myChuck.SetFloat("amp", .5f);
        isTriggered = true;
```

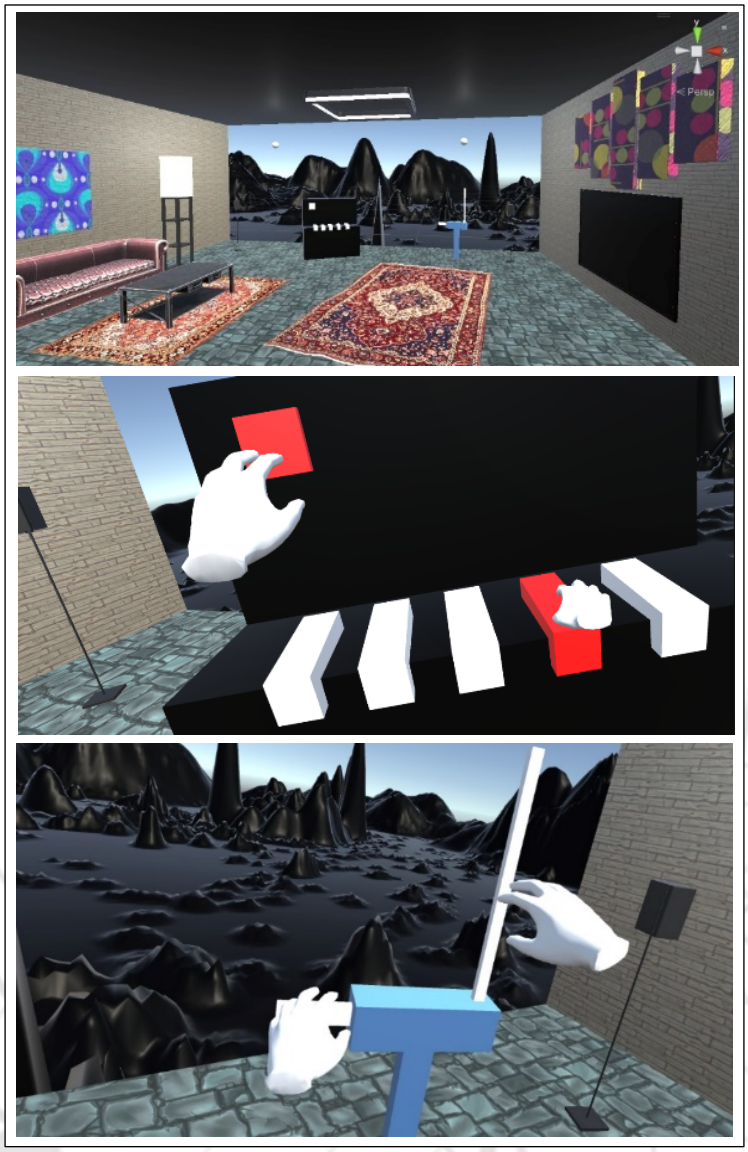
[22] https://docs.unity3d.com/Manual/CollidersOverview.html
[23] Bu örnekteki *ChucK* kodu, *ChucK* dökümantasyonunda yer alan interpolate.ck örneği uyarlanarak oluşturulmuştur. https://chuck.stanford.edu/doc/examples/vector/interpolate.ck

```
    }
}

private void OnCollisionExit(Collision collision)
{
    myChuck.SetFloat("amp", 0);
    isTriggered = false;
}
```

Etkileşimli Müzik Odası

Etkileşimli Müzik Odası (EMO), kullanıcının parametrelerini çarpışma, konum ve mesafe bilgileri aracılığıyla kontrol edilebildiği iki elektronik çalgının yer aldığı, üç tarafı kapalı bir oda biçiminde tasarlanmış, açık yüzü post apokaliptik görünümlü bir manzaraya bakan sanal bir mekândır. SG desteğine de sahip bu tasarımda kullanıcı, mekân içerisinde *Oculus Quest 2* sanal gerçeklik gözlüğü ve kontrolcülerini kullanarak gezinebilmekte, hatta oda dışında süzülebilmekte ve çalgılar ile etkileşime geçebilmektedir. EMO, İstanbul Kalkınma Ajansı destekli TR/10/12/YES/0028 numaralı proje kapsamında Yıldız Teknik Üniversitesi Sanat ve Tasarım Fakültesi'nde kurulan IMC Laboratuvarının teknik altyapısı kullanılarak tarafımızdan hazırlanmıştır.



Şekil 7: Tasarlanan SG ortamı ve çalgılar

Odada bulunan çalgılardan ilki, tuşlarına dokunularak bir oktavlık pentatonik dizinin seslendirilebildiği bir elektronik klavye biçiminde tasarlanmıştır. Ses üretme mekanizmasının merkezindeki ChucK kodu, çalgıya bileşen olarak eklenen *SynthController* adlı bir betik içerisinde yer almaktadır. Kullanıcı elleri ile tuşlara dokunduğunda, her bir tuşa eklenmiş olan *KeyController* betiği içerisinde çarpışma denetlenir ve ilgili tuşun adı *SynthController* betiği ile paylaşılır, ardından tuş adına bağlı olarak MIDI tuş numarası *ChucK* koduna iletilir. *SynthController* betiğine ait *Update()* fonksiyonu aşağıda görülmektedir.

```
private void Update()
{
   switch(keyName)
   {
      case "Key_1":
         note = 60;
         break;
```

```
            case "Key_2":
                note = 62;
                break;
            case "Key_3":
                note = 64;
                break;
            case "Key_4":
                note = 67;
                break;
            case "Key_5":
                note = 69;
                break;
            default:
                break;
        }
        Synth.SetFloat("note", note);
    }
```

Kullanıcının tuşlara dokunuşu (çarpışma) ise *KeyController* betiğine ait *OnCollisionEnter()* ve *OnCollisionExit()* fonksiyonları içerisinde aşağıda görüldüğü şekilde denetlenmekte ve temas edilen tuşun adı ile genlik değeri, *triggerNote()* fonksiyonu aracılığıyla *SynthController* betiği ile paylaşılmaktadır.

```
    private void OnCollisionEnter(Collision collision)
    {
        if (lisCollided && collision.collider.tag == "contacter")
        {
            isCollided = true;
            triggerNote(this.name, 0.4f);
        }

    }

    private void OnCollisionExit(Collision collision)
    {
        isCollided = false;
        _accessToSynthController.ampl = 0f;
        _accessToSynthController.ampl = 0;
    }

    private void triggerNote(string keyName, float amp)
    {
        _accessToSynthController.keyName = keyName;
        _accessToSynthController.ampl = amp;
    }
```

İkinci çalgı ise basit bir Theremin biçiminde tasarlanmış olup, sesin genliği ve perdesi çalgının sol ve sağ tarafında bulunan antenler ile kullanıcının elleri arasındaki uzaklıklara (konumlarına) bağlı olarak kontrol edilebilmektedir. Bu denetleme işlemi çalgıya eklenmiş *ThereminController* betiğine ait *update()* fonksiyonu içerisinde gerçekleşmekte ve hesaplanan konum-mesafe bilgileri normalleştirildikten sonra *ChucK* koduna frekans ve genlik değeri olarak aktarılmaktadır.

```
void Update()
{
   Vector3 freqAntPos = freqAnt.transform.position;
   Vector3 rHandPos = rightHand.transform.position;
   freqDistance = Vector3.Distance(freqAntPos, rHandPos);

   Vector3 ampAntPos = ampAnt.transform.position;
   Vector3 lHandPos = leftHand.transform.position;
   ampDistance = Vector3.Distance(ampAntPos, lHandPos);

   float f = ((1f - freqDistance) * 2000) + 100;
   float a = (0.5f - ampDistance);
   if (a < 0) a = 0;

   if (freqDistance < 1) {
      myTheremin.SetFloat("frequency", f);
      myTheremin.SetFloat("ampTheremin", a);
   }
   else
   {
      myTheremin.SetFloat("frequency", 0);
      myTheremin.SetFloat("ampTheremin", 0);
   }
}
```

Sonuç

Eğer konu teknolojiyse en sık işitilen cümlelerden biri şudur: "Geçmişte de yoktu ama sonra oldu!". Genelde SG, özelde *metaverse* için de durum aynen böyle. Kullanıcı SG gözlüğünü takacak, her türden dijital malzemesini üzerine giyecek ve her yöne hareket edebilen bir aygıt üzerine çıkıp kendini zamanın aktığı, avatarlardan geçilmeyen, paranın kullanıldığı sanal bir dünyanın içinde bulacak. Buna söylenecek tek bir şey var: "Geçmişte yoktu…". Peki şu anda mevcut mu? Kısaca hayır, halen bekliyoruz…

Böylesine bir beklenti her alanda olduğu gibi ses-müzik dünyasında da var. Hatta birçok alana göre belki biraz daha hazırlar. Justin Bieber, Ariana Grande gibi müzisyenlerle Universal Music gibi dev prodüksiyon şirketleri çoktan yola çıktılar bile. Ancak sanal bir ortamda herkesi tatmin edebilecek tam bir müzik performansı günümüz teknolojisinde henüz çok erken bir beklenti. Üstelik buna bir de canlı çalım çalgı performansları eklendiğinde beklenti süresini birkaç yıl daha ötelemek gerek. Peki müzik için SG çok mu gerekli? Eğitimi, bilimi, teorisi vs. için belki evet ama konser gibi performansa dayalı etkinliklerde biraz şüpheliyiz. Örneğin Adorno'ya göre bir konser ortamı, belirli kurallara uyanlarla yalnızca eğlenenler arasındaki geniş bir yelpazede yer alır ve ona göre esas olan kültürün tüketimidir (Adorno, 1977). Konser, üretim ve tüketimin birlikte yapıldığı yoğun kültürel bir ortamdır ve müzik yapanlar, dinler kitle ve arasındaki her türden etkileşimle birlikte mekânın kendisiyle, çevredekilerle ve hatta kokusuyla tam bir bütündür (Güven, 2022). Bu açıdan bakıldığında günün

birinde *metaverse* ortamda konser olsa bile gerekliliği konusunda tartışmaların devam edeceği kesin. Ancak kültür bir tarafa yalnızca teknik olarak bakılacak olsa bile tam bir *metaverse* ortamdan bahsetmek günümüzde halen çok erken. Sadece düşük internet hızları ve yüksek *latency* değerlerinin bir hayli güncellenmesi gerektiğini söylemek bile bizi nasıl bir sürecin beklediğini gösterir nitelikte…

Bu çalışmada, bilgisayar ortamında ses sentezleme ve analizi amacıyla geliştirilmiş bir programlama dili olan *ChucK* ile günümüzün popüler oyun geliştirme ortamlarından bir tanesi olan *Unity* arasında doğrudan iletişimi sağlayan *Chunity* eklentisi aracılığıyla, sanal bir ortamda nesneler ile kullanıcı arasındaki etkileşimin müzikal parametrelere ne şekilde yönlendirilebileceğine ilişkin yöntemler değerlendirilmiş olup, bu yöntemlerden faydalanılarak bir SG gözlüğü ve kontrolcüler aracılığıyla kullanıcının etkileşime geçebileceği iki elektronik çalgı tasarlanmış ve bir sanal ortam içerisine yerleştirilmiştir. Araştırma, tasarım ve uygulama sürecindeki deneyim ve gözlemlerden elde edilen bulgulara yönelik sonuçlar aşağıda sunulmuştur:

- ✓ Oyun motorları; sundukları görsel tasarım, fiziksel motor, ses desteği, kullanıcı etkileşimi ve çok kullanıcılı uygulama geliştirmeye olanak sağlayan araçları bir arada sunan, yalnızca oyun dünyasının değil, oyun-dışı sanal uygulamaların da geliştirilebileceği son derece kullanışlı SG amaçlı araçlardır.

- ✓ Ses, en az görsel kadar kullanıcının sanal ortam içinde yer alma algısını güçlendiren önemli unsurlardan biridir. Oyun motorları bu desteği kaynak-alıcı ilişkisi içinde tasarımcıya sunar. Ses nesneleri, örneklenmiş ses dosyaları biçimindeki ses kliplerinden meydana gelebileceği gibi, programlamaya dayalı bir teknik olan yordamsal ses aracılığıyla ortamın dinamiklerine ve nesneler arası etkileşime bağlı olarak gerçek zamanlı tepki verebilecek nesneler biçiminde de tasarlanabilir. Bu anlamda yordamsal ses, özellikle bilgisayar tabanlı elektronik müzik üretimi bağlamında perde, gürlük ve tını gibi parametrelerin kullanıcı ile ortam arasındaki etkileşime bağlı olarak denetlenebilmesi amacıyla oldukça kullanışlı bir yöntemdir.

- ✓ Temelde birer tasarım ve programlama ortamı olan oyun motorları doğaları gereği yordamsal ses tasarımına olanak sağlamakla birlikte, oyun motorları için geliştirilmiş özel eklentiler ses sentezlemeye yönelik geliştirilmiş programlama dillerinin oyun motoru içerisine doğrudan dahil edilebilmesini mümkün kılar. Bu sayede tasarımcının üzerindeki kodlama yükü hafifler. Bu çalışma sürecinde, birden fazla programlama dilinin *Unity* oyun motoru içerisinde bir arada çalışabildiği ve aralarında veri iletiminin mümkün olduğu gözlemlenmiştir. Ses kliplerinin de uygulamaya dahil edilebildiği göz önünde bulundurulduğunda, bu şekilde bir kullanımın tasarımcıya her dilin kendine özgü en güçlü özelliklerinden aynı anda yararlanabilme olanağı sağlayacağı açıktır.

- ✓ Oyun motorlarının sunduğu görsel tasarım olanakları ile günümüz sanal gerçeklik teknolojileri, gerçek dünya benzetimlerinden yalnızca tasarımcının hayal gücü ile sınırlı özgün yeni dünyalar yaratmaya imkân tanıyacak bir seviyeye ulaşmış olup büyük bir hızla gelişimini

sürdürmektedir. Bu tasarımlar, bilgisayar tabanlı elektronik müziğin sağlayacağı performans olanakları ve tını zenginliği ile bir araya getirildiğinde, sanal gerçeklik ortamında gerçekleştirilecek bir müzikal performansın, müzisyene ve dinler kitleye gerçek dünyada olduğundan çok daha farklı bir deneyim yaşatacağını söylemek pek de yanlış olmaz. Yani kısaca, Adorno'nun kültür tüketimi tespiti gerçek dünya için geçerli olsa da sanal bir dünyada yapılacak performansın o dünyanın kurallarıyla uygulanabilir olması kaçınılmazdır. Bu durumda farklı bir evren ve hatta farklı bir kültür ortaya çıkar. Dolayısıyla bu evrenin insana kültürel etkisi de farklı olacaktır.

KAYNAKÇA

Adorno, T. & M. Horkheimer. (1977). The Culture Industry: Enlightenment as Mass Deception. In Mass Communication and Society (edt: J. Curran and M. Gurevitch). London: *Critical Essays on Consumer Culture*, Westport, CT: Praeger.

Bayraktar, E., Kaleli, F. (2007). Sanal Gerçeklik ve Uygulama Alanları. Akademik Bilişim Konferansları, Dumlupınar Üniversitesi, Kütahya, 31 Ocak-2 Şubat 2007, http://t.ly/_ZH8 (Erişim Tarihi: 15.08.2022)

Brennan, S. (2009). "Redefining MMOs: Pesky Persistence", http://www.engadget.com/2009/08/14/redefining-mmos-pesky-persistence/ (Erişim Tarihi: 15.08.2022)

Cáceres, J. P., Hamilton, R., Iyer, D., Chafe, C., & Wang, G. (2008). To The Edge With China: Explorations in Network Performance. In ARTECH 2008: Proc. of the 4th International Conference on Digital Arts, pp. 61-66.

Çalcı, S. (2014). Gerçeklik Anonim Bir Süreçtir: Whitehead Ontolojisinde Dinamizmin Kuruluşu Üzerine Bir İnceleme. Felsefe ve Sosyal Bilimler Dergisi (FLSF), Sayı 17, s.197-214.

Farnell, A. (2007). An Introduction to Procedural Audio and its Application in Computer Games. In Audio Mostly Conference, Vol. 23, pp. 1-31.

Goldstone, W. (2009). Unity Game Development Essentials. Packt Publishing Ltd.

Gregory, J. (2018). Game Engine Architecture. (3th Edition). A K Peters/CRC Press.

Güven, U. Z. (2022). Kentten Sesler: Metaverse, NFT ve Müzikte Dönüşüm, Sanattan Yansımalar, https://www.sanattanyansimalar.com/yazarlar/ugur-zeynep-guven/metaverse-nft-ve-muzikte-donusum/2724/ (Erişim Tarihi: 15.08.2022)

Huizinga, J. (1938). Homo Ludens: Oyunun Toplumsal İşlevi Üzerine Bir Deneme (Çevr: Mehmet Ali Kılıçbay, Türkçe 2. Baskı, 2006), Ayrıntı Yayınları.

Kaltenbrunner, M., Jorda, S., Geiger, G., & Alonso, M. (2006). The Reactable*: A Collaborative Musical Instrument. In 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06), pp. 406-411.

Kapur, A., Cook, P. R., Salazar, S., & Wang, G. (2015). Programming for Musicians and Digital Artists: Creating Music with ChucK. Manning Publications Co.

Keene, S. (2019). Google Daydream SG Cookbook: Building Games and Apps with Google Daydream and Unity. Addison-Wesley Professional Press.

Koçyiğit, S., Tuğluk, M. N., Kök, M. (2007). Çocuğun Gelişim Sürecinde Eğitsel Bir Etkinlik Olarak Oyun. Atatürk Üniversitesi Kazım Karabekir Eğitim Fakültesi Dergisi, Sayı: 16, s. 324-342.

Linowes, J. (2015). Unity Virtual Reality Projects. Packt Publishing Ltd.

Nevelsteen, K. J. (2018). Virtual World, Defined From A Technological Perspective and Applied to Video Games, Mixed Reality, and The Metaverse. Computer Animation and Virtual Worlds, Vol.29 (1), e1752.

Saridakis, E. (2016). Information, Reality, and Modern Physics. International Studies in the Philosophy of Science, V 30(4), pp. 327-341.

Sinclair, J. L. (2020). Principles of Game Audio and Sound Design: Sound Design and Audio Implementation For Interactive and Immersive Media. CRC Press.

Singhal, S.&Zyda, M. (1999). Networked Virtual Environments: Design and Implementation. Addison-Wesley Professional Publication.

Sherman, W. R., & Craig, A. B. (2018). Understanding Virtual Reality: Interface, Application, and Design. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, Series Editor: Brian A. Barsky, University of California, Berkeley

Spence, J. (2008). Demographics of Virtual Worlds. Journal For Virtual Worlds Research V 1(2), https://jvwr-ojs-utexas.tdl.org/jvwr/index.php/jvwr/article/view/360 (Erişim Tarihi: 15.08.2022)

Stephenson, N. (1992). Snow Crash (1993 paperback ed.), New York: Bantam Books. p. 440 (Tr: Parazit, Çvr: Sibel Hacıoğlu, Altıkırkbeş Yayınları, 2016).

Şekerci, C. (2017), Sanal Gerçeklik Kavramının Tarihçesi, Uluslararası Sosyal Araştırmalar Dergisi (ASOS Journal), Cilt 10, Sayı 54, V.10, s. 1126-1133