BAHÇEŞEHİR ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BİLGİ TEKNOLOJİLERİ PROGRAMI

KAMERA KAYNAKLI İŞİTSEL-SANAT ARACI ARAYÜZÜ GELİŞTİRİLMESİ

Yüksek Lisans Tezi

Serkan ŞİMŞEK
(Lisans: Yıldız Teknik Üniversitesi, İletişim Tasarımı Bölümü)

Tez Danışmanı: Doç. Dr. Adem KARAHOCA

İSTANBUL, 2008

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
BİLGİ TEKNOLOJİLERİ YÜKSEK LİSANS PROGRAMI

Tezin Adı: Kamera Kaynaklı İşitsel-Sanat Aracı Arayüzü Geliştirilmesi
Öğrencinin Adı Soyadı: Serkan Şimşek
Tez Savunma Tarihi: 15.05.2008

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğu Enstitümüz tarafından onaylanmıştır.

Prof. Dr. Erol Sezer
Enstitü Müdürü

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğunu onaylarım.

Yrd. Doç. Dr. Orhan GÖKÇÖL
Program Koordinatörü

Bu Tez tarafımızca okunmuş, nitelik ve içerik açısından bir Yüksek Lisans tezi olarak yeterli görülmüş ve kabul edilmiştir.

Jüri Üyeleri	İmzalar
Doç. Dr. Adem KARAHOCA (Danışman) Bahçeşehir Üniversitesi Mühendislik Fakültesi	
Prof. Dr. Nizamettin AYDIN Bahçeşehir Üniversitesi Mühendislik Fakültesi	
Yrd. Doç.Dr. Yalçın ÇEKİÇ Bahçeşehir Üniversitesi Mühendislik Fakültesi	

ÖNSÖZ

"Kamera Kaynaklı İşitsel-Sanat Aracı Arayüzü Geliştirilmesi" konulu bu tez çalışması Bahçeşehir Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Bilgi Teknolojileri Yüksek Lisans Programı'nda hazırlanmıştır.

Tez çalışmamız boyunca geliştirdiğimiz bu araç, ses ve görüntü etkileşimi konusunda yapacağımız akademik çalışmaların başlangıcını oluşturmaktadır. Uzun bir yol olarak görülen etkileşimli ortam tasarımının derinlerine indikçe, yapacaklarımızın heyecanını daha yoğun hissediyorum. Yüksek lisans tezimin bu son noktasında; çalışmamızın, artısı ve eksisiyle bize ait olan özgün bir eser olmasının memnuniyetini yaşamaktayım.

Yüksek lisans programına başvurduğum günden bugüne kadar yanımda olan, tez sürecinde disiplinler arası nitelikte özgün bir çalışma yapmam konusunda benden desteğini hiçbir zaman esirgemeyen ve tavırlarıyla cesaretlendiren, her konuda bana olan güvenini hissettirerek motivasyonumu yüksek tutmamı sağlayan danışmanım Doç. Dr. Adem Karahoca'ya yüksek lisans eğitimim süresince yaptığı her şey için çok teşekkür ederim.

Ayrıca; seslerin ve renklerin fiziksel özelliklerini araştırmam ve kaynaklara ulaşmam konusunda yardımlarını esirgemeyen İstanbul Teknik Üniversitesi Fizik Bölümü Araştırma Görevlisi Muzaffer Erdoğan'a yardımlarından dolayı teşekkürü borç bilirim.

Ve, her zaman olduğu gibi bu çalışmamda da yanımda olan, karşılıksız sevgileri ve fedakarlıkları ile kendimi hep şanslı hissetmemi sağlayan, bugüne gelmem de emeklerinin karşılığını asla ödeyemeyeceğim aileme, tez çalışmam süresince verdikleri manevi destekten dolayı sonsuz teşekkürlerimi sunarım.

Mayıs, 2008
Serkan Şimşek

İÇİNDEKİLER

ŞEKİLLER

TABLOLAR

KISALTMALAR

API : Uygulama programlama arayüzü

Hz : Hertz (frekans birimi)

Nm : Nanometre

THz : Terahertz

USB : Universal Serial Bus (evrensel seri veriyolu)

SEMBOLLER

$ **:** Amerika Birleşik Devletleri'nin resmi para birimi

 **:** Programlama dilinde ‘for’ döngüsü

 **:** Programlama dilinde ‘if’ sınaması

 **:** Programlama dilinde ‘switch’ sınaması

 **:** Elektronik devrelerde ‘veya’ kapısı

ÖZET

KAMERA KAYNAKLI İŞİTSEL-SANAT ARACI ARAYÜZÜ GELİŞTİRİLMESİ

Şimşek, Serkan

Bilgisayar Mühendisliği Anabilim Dalı
Bilgi Teknolojileri Yüksek Lisans Programı

Tez Danışmanı: Doç. Dr. Adem Karahoca

Mayıs, 2008, 216 sayfa

Bu tez çalışması, kamera kaynaklı işitsel-sanat (sound-art) aracı ve arayüzü tasarımının önerisi ve uygulamasından oluşmaktadır.

Disiplinler arası bir düşünceyle, görüntü ve ses ortamlarının etkileşim imkanları incelendi ve çıkarılan sonuçlarla bir yazılım oluşturuldu. Bu yazılımın kullanıcıları, işitsel-sanatın deneysel doğasında yeni anlatım imkanları arayan sanatçılar ve icracılardır.

Günümüzde dijital teknolojilerin sınırsız gelişmesi, görsel ve duysal ortamlar arasında ilişkiler kurulmasıyla oluşturulan etkileşimli çalışmaların imkanlarının genişlemesini doğurmuştur. Bu doğrultuda hareket edilerek, tarafımızdan, kameranın bir ses enstrümanı olarak kullanılmasını ve piksellerin ses frekanslarına dönüşmesini sağlayan bir yazılım geliştirildi. Görüntü işleme ve dijital ses üretiminin kullanıldığı bu yazılım, Processing ortamında Java programlama dili ile oluşturuldu.

Çalışmamız dört bölümden oluşmaktadır. Bunlar; giriş, malzeme ve yöntem, bulgular ve sonuç ve yorumlardır.

Giriş bölümünde; tezin temel motivasyonu, işitsel-sanatın ne olduğu, fikrimize arka plan oluşturan ve esin kaynağı olan çalışmalar ve bunların değerlendirilmesi ile belirlenen hedefler incelendi.

Malzeme ve yöntem bölümünde; öncelikle çalışmamızın malzemesini oluşturan ses ve rengin fiziksel özellikleri, ardından analog ve dijital teknolojiler, hoparlörler ve kameralar araştırıldı. Yöntem olarak, seçtiğimiz yazılım geliştirme ortamı ve kullandığımız programlama dilinin özellikleri incelendi. Etkileşim tasarımı olarak ses ve görüntüyü nasıl eşleyeceğimiz belirlendi.

Bulgular bölümünde geliştirilen yazılımın ayrıntıları ele alındı. Aracın temel özellikleri ve kullanıcılarına neler sunduğu, aracın temel yapısı, sistemin çalışma şekli ve algoritmaları incelendikten sonra, arayüzün temel işlev ve özellikleri ortaya konuldu.

Tezin son aşaması olan sonuç ve yorumlar bölümünde, tez süreci değerlendirildi ve gelecek çalışmalara yönelik belirlemeler de bulunuldu.

Anahtar Kelimeler: etkileşim tasarımı, görüntü işleme, dijital ses üretimi, işitsel-sanat, ses ve görüntü eşleşmesi, grafik kullanıcı arayüzü

ABSTRACT

DEVELOPING A CAMERA SOURCED SOUND-ART TOOL INTERFACE

Şimşek, Serkan

Department of Computer Engineering
Information Technologies Graduate Program

Thesis Advisor: Assoc. Prof. Dr. Adem Karahoca

May, 2008, 216 pages

This thesis consists of a proposition and its application of a camera sourced sound-art tool interface.

Within a multi-disciplinary perspective, firstly the interactional possibilities of visual and sound media is analyzed and afterwards software has been developed. The intended users of this software are sound-artists and performers who seek new ways of expression through the experimental nature of sound-art.

The boundless developments of today's digital technologies in the recent times have introduced extended possibilities concerning the relation of visual and sound media based interactive studies. Within this scope, software which provides the usage of camera as a sound instrument and thus enables the transformation of pixels into sound frequencies has been developed. The software using image processing and digital sound generation is developed in Processing by the Java programming language.

This study composed of four chapters. These are; introduction, material and method, findings and conclusion/comments.

In the introductory chapter; the main motivation of the thesis, the meaning of sound-art, the previous inspiratory studies which form the background to our ideas and the goals determined after the evaluation of these studies are analyzed.

In the material and method chapter; firstly the physical characteristics of sound and color which constitute the main materials of our study and then analog and digital technologies, loudspeakers, cameras are researched. As the method, the chosen software development environment and the programming language are analyzed. In order to design interaction, the ways to match the sound and visual has been designated.

In the findings chapter, the details of the developed software are discussed. After the main features and user-benefits of the tool along with its main structure, working procedure and algorithms are analyzed, the main functions and features of the interface have been put forth.

In the last chapter, conclusion and comments, the process of the thesis is evaluated and suggestions for future studies are made.

Keywords: interaction design, image processing, digital sound generation, sound-art, sound image relationships, graphic user interface

1. GİRİŞ

"Böylece, yolun farklı noktalarında bulunan sanat dallarının her biri, ifade etmekte en başarılı olduğu şeyi, kendine özgü bir dille ortaya koymaktadır. Aralarındaki farklılıklara rağmen –ya da belki de bu farklılıklar sayesinde-, sanat dallarının, bugün, ruhsal yaşantının bu geç safhasında olduğu denli birbirine yaklaştığı bir dönem olmamıştır.

Bir sanat dalının diğerinden metod ödünç alışı, ancak, ödünç alınan metod yüzeysel bir biçimde değil, temel olarak kullanıldığında başarıya ulaşabilir. Bir sanat dalı, öncelikle diğer sanat dallarının metodlarını nasıl kullandığını öğrenmelidir. Böylelikle bu metodlar, ödünç alanın sanatına uygun bir şekilde aktarılabilir. Sanatçı, her metodu doğru uygulama gücüne sahip olduğunu, fakat bu gücü geliştirmesi gerektiğini unutmamalıdır." (Kandinsky, 1911)

1.1 MOTİVASYON

Günümüzde farklı ortamlara ait verilerin değerlendirilip yaratıcı bir bireşim ile yeniden üretildiğine birçok sefer tanık oluyoruz. Bir ortamda bulunan fiziksel özellikler, başka ortamların özelikleri ile eşlenerek ya da bazı sayısal işlemler sonucunda farklı sonuçlar elde edilerek, birden çok ortam etkileşimli hale getirilebiliyor. Bu ortamların en belirgin olan iki tanesi ses ve görüntü ortamlarıdır. Ses ve görüntü, özellikleri farklı olan iki ortamdır. Bunlar tarihte dönem dönem karşılaştırmalı olarak birçok sefer irdelenmiş ve sanatsal olarak değerlendirilmeye çalışılmıştır. Bu tezin amacı da, bu alanların özellikleri arasında belli sayısal işlemler ile oluşturulacak ilişkiler yaratmak ve kamera kaynaklı görsel verilerin özellikleri ile ses ortamının özelliklerini belirleyerek bir işitsel-sanat (sound-art) aracı tasarlayabilmektir. Bu araç, belli bir arayüz ile ses ve görüntünün nasıl karşılaştırılabileceğini ve oluşturulan seslerin özelliklerinin nasıl kontrol edilebileceğinin imkanlarını, kullanıcısı olan işitsel-sanat icracılarına (sound-artist) sunacaktır. Özellikle işitsel-sanatın her türlü özgün ve deneysel çalışmaya açık olan

doğası düşünüldüğünde, görüntülerden üretilen parametrelerin ses özelliklerini belirlemesi, yani kameranın bir ses enstrümanı olarak kullanılması ve bunun parametre kontrollerinin yapılacağı bir arayüzün tasarlanması, işitsel-sanat icracılarının yeni denemelerine katkıda bulunacak bir zemin oluşturacaktır.

Hedeflenen amaç, tasarlanan işitsel-sanat aracının sistem yapısının oluşturulması, temel etkileşimlerinin belirlenmesi ve sistemin çalıştırılmasıdır. Oluşturulan sistemin en temel özelliği; kullanıcının, ses ve görüntü ilişkisi zemininde girebileceği etkileşimlerin tartışılması ile oluşturulmuş olmasıdır. Tasarlanan işitsel-sanat aracının ve arayüzünün, hedef araştırmalar doğrultusunda düzeneği oluşturulmuş ve teknoloji olarak işlerliği somutlanmaya çalışılmıştır. Asıl motivasyonumuz, işitsel-sanat aracını etkileşimli bir düzenek olarak ortaya koymak ve bilgi teknolojileri açısından mümkün kılmak olmuştur. Çalışmamızda vurgumuz, etkileşim tasarımı kararlarımızın uygulanabilmesi, oluşturduğumuz kodların performansı, sistemin üzerine oturduğu veri modelleri ve işlevsel mimari ve bunların uygulanabilirliği konuları üzerindedir. Bu noktadan sonra; grafik arayüz tasarımı için kavramsal bir çerçevenin çizilmesi ve bunun uygulanması, insan bilgisayar etkileşimi alanına yönelik olarak kullanıcı davranışları ile ilgili saptamaların yapılması, makine-sistem öğrenilebilirliğinin tartışılması ve bu araç ile üretilen performansların estetik olarak analiz edilerek sınıflandırılması daha sonraki araştırmalarımızın konuları olacaktır.

Bu çalışmamız, ses ve görüntü eşleşmesi üzerine yapılmış bilgisayar destekli olan ya da olmayan ilk çoklu ortam çalışması değildir. Amaç, daha önce yapılmış olan çalışmaları incelemek, bunlar ile hedefe yönelik ortak noktalar belirlemek ve çıkartılan sonuçlar doğrultusunda özgün öneriler de bulunarak, bir kamera kaynaklı işitsel-sanat aracının teorik ve pratik temellerini oluşturmaktır.

1.2 İŞİTSEL-SANAT

İşitsel-sanat, sesin ve duyma duyusunun oldukça geniş olan unsurlarını çalışmalarının merkezine alan çeşitli sanat uygulamalarının oluşturduğu bir kategoridir. Daha sonrasında, sanatsal ve algısal alanda sesin ve görüntünün ilişkisinin geliştirilmesi işitsel-sanatçıların çeşitli çalışmalarına konu olmuştur. Çağdaş sanatın birçok türü gibi, işitsel-sanat da disiplinler arası bir doğaya sahiptir ve melez şekiller alabilmektedir. Hemen hemen çağdaş sanatın bütün ilgi alanlarının, akustik, psiko-akustik, elektronik, ses medyası ve ses teknolojisi (analog ve dijital), üretilmiş olan ya da çevreye ait sesler ile insan vücuduna, heykele, filme ya da videoya yönelik olan yeni arayışlar ile ilişki içindedir [1].

Yüzyılın başlarında hemen hemen bütün sanatlarda ortaya çıkan yeni ifade arayışları, özellikle İkinci Dünya Savaşı'nın sonrasında sanatların kendi köklerine dair eleştirelliğe dönüşmüş ve bu durum sanatların tanımlarının yeniden tartışılması sürecini getirmiştir. Sanatın ve sanat eserlerinin klasik varoluş koşullarının sarsılmaya başladığı bu dönemde, işitsel öğeler de bu sorgulama sürecinin içerisine girmiştir.

Özellikle 1980'lerin başlarından itibaren, görsel sanatlarla ilgili alanlarda artan sayıda ses ile ilişkili sanat çalışması yapılmıştır. İşitsel-sanat kapsamında düşünebileceğimiz bu çalışmalarda genelde, müzik, kinetik heykel, çevresel faktörler ya da izleyici tarafından aktive edilen enstrümanlar, kavramsal sanat nesneleri, ses efektleri, ses üreten elektronik devreler ve etkileşimli bilgisayar programları vb. gibi elemanlar kullanılmaya başlanmıştır. Böylece, işitsel-sanat, bir şekilde ses içeren ya da ses üretilen birçok çalışmanın ait olduğu kategoriyi göstermek için kullanılır olmuştur [2].

Bu alanda yapılan çalışmalara verilen işitsel-sanat isminin geçerliliği konusunda tartışmalar da olmuştur. Önemli bir müzik kariyerinin ardından deneysel çalışmalar yapmaya başlayan Max Neuhaus, bir serginin açılış konuşmasında, sanatın icra edildiği ortamın mesajın iletilmesi için yeterli olamayacağını ve çelikten yapılmış heykellere çelik-sanatı denemeyeceği gibi, sesin kullanıldığı çalışmaları anlatmak için de "sound" kelimesinin kullanılmasının bu alanda üretilen eserleri kuşatan bir tanımlama olmaya

yetmeyeceğini belirterek, işitsel-sanat isminin sorgulanması gerektiğini vurgulamıştır [2].

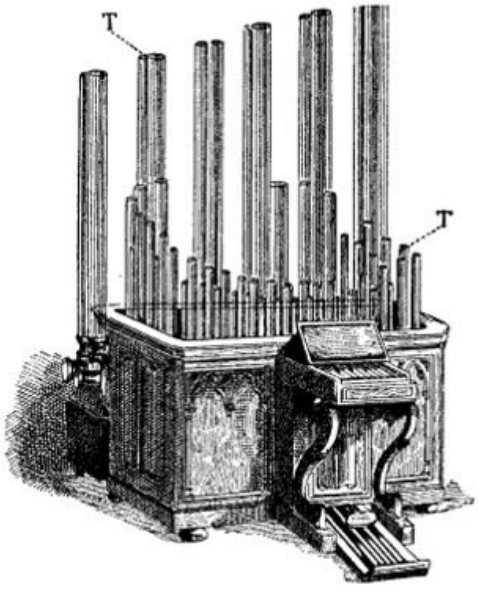
Ancak günümüzde, işitsel-sanatın sadece ses ile değil; görüntü, mekan, hareket vb. gibi alanlarla birlikte düşünülmesi gerektiği ve sınırlarının klasik sanatlarda olduğu gibi net ve katı bir şekilde çizilmesi yerine, disiplinler arası yapısının vurgulanması yönünde uzlaşma sağlanmıştır. Özellikle izleyicinin konumunun da, öncelik verilen müdahale alanlarından birini oluşturan işitsel-sanat, 20.yüzyıla özgü bir sanat şekli olmuştur. Elektronik aletlerin ucuzlaması da bu alanda üretilen deneysel ürünlerin sayısının artmasına yol açmıştır. Böylece, üretim için büyük stüdyo gerekleri ortadan kalkmış ve kişisel bilgisayarlar ve ilgili yazılımlar ile deneysel çalışmaların önü açılmıştır. İşitsel-sanat, bir yandan da geçtiğimiz yüzyıl sanatında önemli bir yer tutan performans ve enstalasyon sanatları ile birlikte düşünülmeye başlanmıştır. Performans ve enstalasyon bağlamında birçok sanatçı işitsel-sanat ile ilgili değişik çalışmalara imza atmıştır. Aynı zamanda akademik dünyada da işitsel-sanat alanı ile ilgili araştırmalar ve deneysel çalışmalar bulunmaktadır (Şenova ve diğ., 2007).

Birçok farklı alanının etkileşimi üzerinde yükselen ve göreceli olarak çok açık tanımlara sahip olmayan işitsel-sanat alanında çalışmalar yapan ve bu nedenle, özgünlükleri ile farklı başlıklar altında da bulunabilecek kişi ve gruplar arasında; Alejandra and Aeron, Miguel Álvarez-Fernández, Maryanne Amacher, Laurie Anderson, John Cage, Janet Cardiff, MW Burns, Lance Dann, Tacita Dean, Disinformation, Daniel Dugas, Judy Dunaway, EMMAX, Brian Eno, Bill Fontana, Bernhard Gal, Brenda Hutchinson, Ryoji Ikeda, Christopher Janney, Miranda July, David Kristian, Christina Kubisch, Bernhard Leitner, Yuri Landman, Arto Lindsay, Annea Lockwood, Alvin Lucier, Christian Marclay, Stephan Mathieu, Abinadi Meza, Christof Migone, George Bures Miller, Paul D. Miller, Meredith Monk, Tracie Morris, Max Neuhaus, Carsten Nicolai, Roberto Paci Dalò, Don Ritter, Pauline Oliveros, Yoko Ono, John Oswald, Nam June Paik, Charlotte Moorman, Norbert Walter Peters, Don Simmons, Soundlab, Karlheinz Stockhausen, Rod Summers, Jeff Talman, Yasunao Tone, Stephen Vitiello, Carl Michael Von Hausswolff, Bill Viola, Hildegard Westerkamp, LaMonte Young isimleri sayılabilir [3].

1.3 ARKA PLAN

Yaptığımız çalışmanın temelinde bulunan ana fikrin oluşmasında esin kaynağı olan ve yol gösterici nitelikte bulunan birçok akademik ve profesyonel çalışma vardır. Bu çalışmaların önemi, kamera kaynaklı işitsel-sanat aracı olarak tanımladığımız kendi önerimizi oluşturma da bizim için farklı yönleri ile esinleyici tecrübeler olmalarındandır. Tasarladığımız aracın fikirsel arka planını oluşturan ve bizim için ufuk açıcı olmuş olan bu çalışmalar, kendi çalışmamız için anlamlı olan yönleri ile incelenecektir.

Görüntü ile ses ortamları arasında birlik sağlanması ve bu ortamlardan en az birinin diğeri üzerinde belirleyici olması uzun dönemlerden beri birçok sanatçının ve bu ortamlarla ilgili deneysel çalışmalar yapan kişilerin ilgi alanında olmuştur. Özellikle ses ortamının görsel etkiler ürettiği ve *"görsel müzik"* (visual music) olarak adlandırılan etkinliği imkanlı kılan mekanizmalar yüzyıllar öncesinden beri üretilmiştir. Bu mekanizmaların ortak özelliği sesin üretimi aşamasında bu seslere karşılık gelen görsel etkileri de aynı anda üretmesidir. Özellikle dinsel etkinlikler sırasında, atmosferin izleyiciler üzerinde daha kuşatıcı olmasını sağlamak amaçlı da üretilmiş olan birçok mekanizmanın en eski olarak bilineni, bir Cizvit papazı ve matematikçi olan Louis-Bertrand Castel tarafından 1734 yılında üretilmiştir. Doğa ve dini mistisizm ile ilgilenen Castel, geleneksel klavsen ile birlikte arkadan renk şeritleriyle aydınlatma yapan bir seri mumu birleştirerek müzikal atmosferi destekleyen ve sesler ile üretilen görüntüler yaratan bir mekanizma tasarlamıştır. Bu tarihten sonra, birçok görsel müzik mekanizması üretilmiştir. Bu üretilen mekanizmalardan birisi olan Pyrophone, 1869 yılında Frederic Kastner tarafından üretilmiştir. (Şekil 1.1)



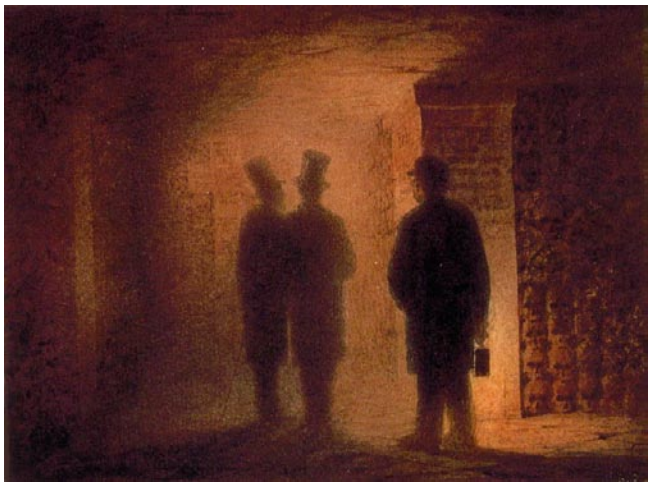
(Kaynak: www.wikipedia.org)
Şekil 1.1 Frederic Kastner tarafından 1869 yılında üretilen Pyrophone

Pyrophone, diğer görsel müzik mekanizmaları gibi ses ve görüntü üretmektedir. Bunun için, içlerinde gazların bulunduğu kristal tüpler kullanılmıştır. Bu tüplerin içlerindeki gazların parlaması ile sesin yanında aynı anda alevler de ortaya çıkmaktadır (Levin, 2000).

Görsel müzik alanında uzun yıllar çalışmalar yapmış olan Tom DeWitt, bir piyanonun müzik için kusursuz bir alet olmasını, görsel harmoni üretmek için de mükemmel bir alet olması yolunda yeterli görmektedir. Özellikle piyano klavyesinin görsel kompozisyon kullanımlarında yeterli bir araç olacağı fikrindedir. Bu yolda yapılan çalışmalar, bir süre sonra bu yöntemi özel bir sanatsal forma dönüştürecektir (Dewitt, 1987).

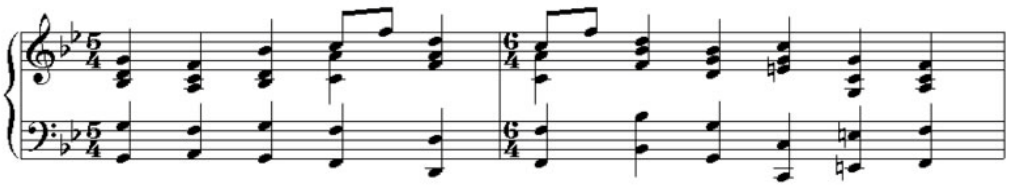
Birbiri ile benzer mekanizmaya sahip olan bu görsel müzik düzenekleri, sistemli olarak ses ile belirlenen görsel etkilerin oluşmasını sağlamaktadır. Bu mekanizmaların, seslerin görüntü ile belirlenmesi amacını taşıyan bizim çalışmamız için önemi, ses ve görüntü arasında bir bağ kurulmaya çalışılmış olmasındandır. Dijital teknolojilerin öncesinde üretilmiş olan bu mekanizmalar, her ne kadar ses ortamından görüntü ortamına doğru gitse de, iki ortam arasında ilişki kurmasıyla bizim için önem taşımaktadır.

Yaptığımız çalışmanın temel özelliği olan görüntüden sese gitme niteliği, kendisine klasik sanatlarda da yer bulmuş olan bir tutumdur. Görsel sanat olarak resimden yola çıkarak müzik eserlerinin üretilmesi, ortak bir tutuma yönelik örnek teşkil etmektedir. Görsel eserlerden yola çıkarak bestelenen müzik eserlerinin en ünlülerinden biri, Modest Mussorgsky tarafından 1874 yılında bestelenen Bir Sergiden Tablolar adlı eserdir [4]. Bu eserde; sanatçı, arkadaşı ressam Victor Hartmann'ın bir sergisinde gördüğü resimlerin etkisinde kalarak, bu resimlerin kendisinde bıraktığı izlenimleri yansıtan bir beste yapmıştır. Böylece besteci, görsel bir bütünlük olan resmi, bir sesler bütünlüğü olan müzik ile ifade etmeye çalışmıştır. (Şekil 1.2 ve Şekil 1.3 ve Şekil 1.4)



(Kaynak: www.wikipedia.org)

Şekil 1.2 ve 1.3 Victor Hartmann'ın sergisinde bulunan resimler



(Kaynak: www.wikipedia.org)

Şekil 1.4 Modest Mussorgsky'nin Bir Sergiden Tablolar adlı eserinden notalar

Burada ortamlar arası dönüşüm, herhangi bir mekanizma ile gerçekleşmemektir. Sadece bir bestecinin kendi duyarlılığı ile görsel eserlerden etkilenerek bir müzik eseri üretmesi söz konusudur. Ancak, bestecinin bu tutumu, resim ile müzik arasında bağ kurulması ve görüntünün sesi belirlemesi dolayımı ile görüntülerden ses üreten aracımızın fikrinin oluşturulması sürecinde bizi besleyen bir esin olmuştur.

Seslerin oluşturulma imkanlarının genişliğini gösteren bir diğer örnek, Theremin adlı elektronik müzik aletidir. Yirminci yüzyılın başlarından itibaren birçok elektronik müzik aleti tasarlanmaya çalışılmıştır. Ancak Theremin'in diğer tasarlanan elektronik müzik aletlerinden farkı, bu aletin dokunulmadan kullanılan ilk müzik aleti olmasıdır. Bu alet, Sovyetler Birliği Hükümeti'nin desteklediği yakınlık sensörleri ile ilgili bir projede çalışan fizikçi Leon Theremin tarafından 1919 yılında icat edilir. Moskova Elektronik Konferansı'ndan alınan olumlu tepkiler sonrasında, Bolşevik lider Vladimir Lenin'in karşısına çıkartılır. Theremin aletinden oldukça etkilenen Lenin, bu aletin kullanımını öğrenebilmek için bir süre eğitim almıştır. Ardından, Theremin'in tüm dünyaya tanıtılması ve Sovyet teknolojisinin geldiği son noktayı göstermesi için, Sovyetler Birliği Hükümeti bu elektronik müzik icadını dünya tanıtım turnesine gönderir. Leon Theremin, 1928 yılında Theremin'in enstrüman olarak patentini alır ve ardından üretim haklarını RCA (Radio Coorperation of America)'ya satar. Bundan sonra Theremin tüm dünyada daha da popülerleşmeye başlamıştır [5].

Bu elektronik müzik aletinin bizim ilgi alanımızda olmasının nedeni, kullanıcı ile olan etkileşim şeklidir. Kullanıcı, bu aleti dokunmadan sadece ellerini yaklaştırarak ve uzaklaştırarak kullanır. Aletin, kullanıcının ellerinin pozisyonuna duyarlı olan iki adet metal anteni vardır. Bu kontrollerden biri ile osilatörlerin frekans değeri, diğeri ile de ses yükseklik değerleri (volume) ayarlanır. Enstrüman devreleri iki adet radyo frekans osilatörü barındırır. Üretilen sinyaller Theremin tarafından yükseltilir ve hoparlörlere gönderilir. (Şekil 1.5)

(Kaynak: www.wikipedia.org)

Şekil 1.5 Theremin adlı enstrüman Leon Theremin tarafından kullanılırken

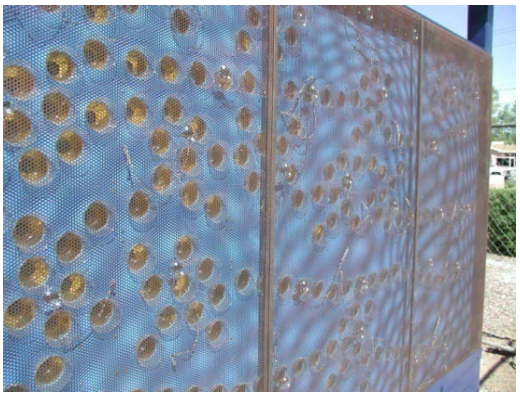
Theremin'in kullanım ilkeleri, yani dokunmadan sadece mesafe ve harekete duyarlı olarak çalışması, sesin enstrüman ile geleneksel bir etkileşim içerisinde bulunulmadan üretilmesi yolunda önemli bir adım olmuştur. Theremin enstrümanı ile sesi görüntülerden yani direk temasın olmadığı bir ortamdan üreten işitsel-sanat aracımız arasında bu etkileşim şekli açısından ortak bir yan bulunmaktadır. Diğer bir nokta da, Leon Theremin tarafından ses özellikleri açısından sadece frekansın değil, aynı zamanda ses yükseklik değerinin de kontrol öğesi olarak seçilmesi bizim aracımız için önemli bir noktadır.

Etkileşim tasarımı açısından, Theremin'in dokunulmadan kullanılan niteliğine benzer bir özellik gösteren Transition Soundings adlı çalışma, etkileşimli işitsel-sanat aracı olarak 2006 yılında gerçekleştirildi (Birchfield ve diğ., 2006). Bu çalışmada öne çıkan fikir, şehrin geçiş yollarından esinlenerek oluşturulacak olan düzeneklerin duraklara yerleştirilmesidir. Halktan insanların kullanımı için tasarlanmış olan bu araç, günlük yaşantı içerisinde durakta bekleyen insanların etkileşimi ile çalışmaktadır. (Şekil 1.6)

(Kaynak: Birchfield ve diğ., 2006)

Şekil 1.6 Transition Soundings adlı ekileşimli işitsel-sanat aracının görünümü

Kullanıcılarında sanat deneyimi aramayan bu etkileşimli düzenek, herkesin kullanımına açık bir mekanizma olarak tasarlanmıştır. Bu şekilde, durakta bulunan herkes aracın potansiyel kullanıcısı olur. Yaklaşık elli adet sensör ve yirmi yedi adet modülden oluşan aracın her bir modülünde, iki yakınlık sensörü, bir ışık sensörü, on adet hoparlör ve bir mikro-kontrol çipi bulunur. Bütün modüller birbirleri ile farklı şekillerde bağlantılara sahiptir. (Şekil 1.7)



(Kaynak: Birchfield ve diğ., 2006)

Şekil 1.7 Transition Soundings adlı çalışmanın mekanizmasının yakından görünümü

Bu çalışma, üretim imkanlarının genişletilmesi için, dört farklı ses tarzını barındırmaktadır. Bu şekilde, kullanıcılar tarafından oluşturulabilen sesler zengin bir yelpazeye kavuşmaktadır. Bizim için bu çalışmanın önemli olması, işitsel-sanat icrası için teknolojik bir düzeneğin farklı tecrübelere imkan tanıyacak şekilde düzenlenmiş olmasından dolayıdır. Herhangi bir müzik eğitimi almamış olmalarına rağmen durakta

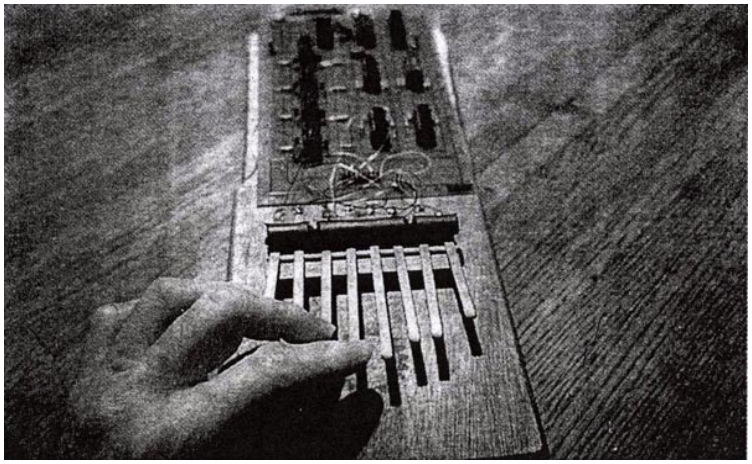
bekleyen insanlardan oluşan kullanıcılar kısa bir süre içerisinde bu araca adapte olabilmektedirler.

Müzik eğitimi almamış olmalarına rağmen, kullanıcılara sesleri belli bir düzenlemeyle oluşturabilme becerisini veren aletlerden bilgisayar tabanlı olanları da vardır. Bilgisayar tabanlı müzik düzenekleri, uzun yıllardan beri kullanılmaktadır ve özellikle 1980'li yıllardan sonra bu konuda yapılmış olan birçok çalışma vardır. Bunlardan bazıları, Paul De Marinis'in 1982 ve 1983 yıllarında gerçekleştirdiği Sound Fountain ve Music Room adlı çalışmalar, George Lewis'in 1986 ve 1987 yıllarında gerçekleştirdiği Mbirascope ve A Map Of The Known World adlı çalışmalar ve David Behrman'ın 1989 yılında gerçekleştirdiği Keys to Your Music adlı çalışmadır (Behrman, 1991). Bu çalışmaların ortak hedefi, bilgisayar teknolojisini kullanarak, herhangi bir müzik enstrümanı çalma tecrübesi olmayan ya da oldukça az olan kullanıcılara müzik yapma deneyimini yaşatmaktır.

Bu beş çalışmanın ortak olan özellikleri arasında;

- Ses üreten enstrümanlar olmaları
- Bilgisayar ile enstrümanlar arasında özel tasarlanmış donanımlar ve yazılımlar olması
- Özel bir yazılım tarafından kontrol edilen bir müzik sistemlerinin olması
- Görsel bilgisayar grafikleri ile kullanıcıları bilgilendirici karakterlerinin olması
- Sisteme eklenmiş olan amfilerinin ve hoparlörlerinin olması

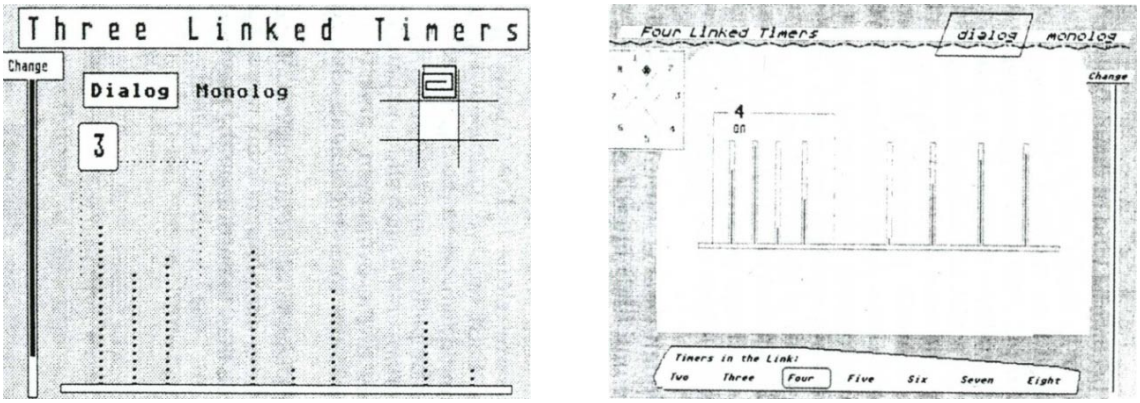
Bu beş çalışmadan en son gerçekleştirileni olan Keys To Your Music adlı çalışma da bu özellikleri göstermektedir. (Şekil 1.8)



(Kaynak: Behrman, 1991)

Şekil 1.8 Bir kullanıcı Keys To Your Music adlı çalışmayı kullanırken

Kullanıcılarına müzik yapma deneyimini yaşatan bu çalışma, galerilerde ve müzelerde aracı kullanmak isteyen kişiler için kendisini hazır bir durumda bekletmektedir. Böylece kullanıcı hiçbir ön hazırlık yapmadan sistemi kullanabilmektedir. Sistem, kullanıcıların tuşlarına dokunması ile ses sonuçları doğurmaktadır. Böylece kullanıcı daha önceden tanıdık geldiği bir tuş kavramına, onun yeniden yorumlanmış şekliyle yeniden yönelmektedir. Kullanıcı aldığı sonuçlara göre, aracla başka bir yönelimle yeniden etkileşime geçecektir. Bu noktada, kullanıcıya yardım edebilmek için grafik arayüz tasarlamıştır. Bu grafik arayüz, kullanıcının, yaptıklarının müzikal olarak ne anlama geldiğini daha iyi anlaması için tasarlanmıştır. (Şekil 1.9 ve Şekil 1.10)



(Kaynak: Behrman, 1991)

Şekil 1.9 ve 1.10 Keys To Your Music adlı çalışmanın kullanıcıya bilgi vermek için tasarlanmış olan grafik arayüzleri

Bu grafik arayüzlerin tasarlanmasında dikkat edilen en önemli kaygı, kullanıcının sistem hakkında bilgi almasının dışında hiçbir eğilime doğru kaymaması olmuştur. Kullanıcı içgüdüsel olarak, yarattığı etkilerin görsel karşılıklarına yoğunlaşma gösterebilir. Bunu engellemek için, bu arayüzler içerisinde oldukça basit ve yalın olan bilgiler verilmiştir. Mekanizma hazırlanırken var olan bu tutum, bizim aracımız için de önem arz eden bir özelliktir.

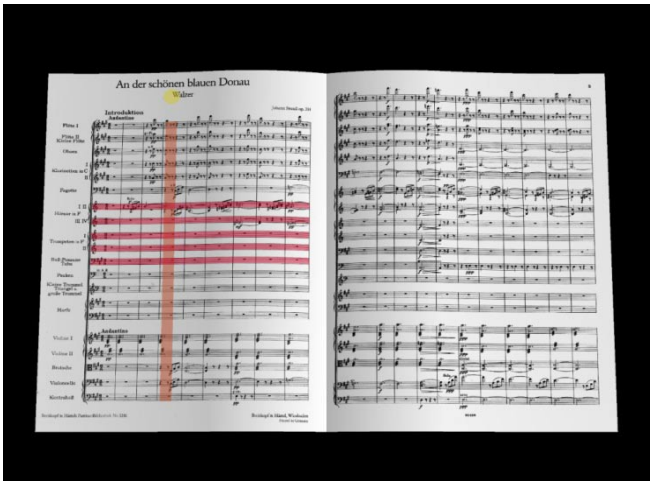
Burada dikkat edilmesi gereken bir nokta da, sistemin ne kadar kullanıcı dostu olacağıdır. David Behrman çalışmasında belirttiği gibi (Behrman, 1991), bu konuda dengede bir tasarım yapılmalıdır. Sistem eğer çok kolay olur ve kullanıcı yerine bütün kararları sistem verirse, kullanıcı herhangi bir şeyi başarma duygusu yaşayamayacaktır. Eğer çok zor olursa, kullanıcı sistemden sıkılacak ve başarıya ulaşamadan uğraşmayı bırakacaktır. Bu yüzden, sistem ne çok kolay, ne de çok zor olmalıdır. Tasarımlar yapılırken ki bu ölçütler, bizim için de önemli olmaktadır.

Dijital teknolojilerin gelişmesi ve bilgisayarların işlem güçlerinin artması kullanıcılara müzik deneyimi yaşatma konusunda daha karmaşık düzenekler tasarlanmasını imkanlı hale getirdi. 2006 yılında yapılan Micon (A Music Stand for Interactive Conducting) adlı çalışma, müzik konusundaki en karmaşık alanlardan biri olan orkestra şefliğini, müzik eğitimi almamış kullanıcıların da deneyimlemesini sağlamaktadır (Borchers ve diğ., 2006). Micon adlı çalışma, etkileşimli orkestra şefliği için tasarlanmış elektronik bir müzik standıdır. (Şekil 1.11)

(Borchers ve diğ., 2006)

Şekil 1.11 Micon müzik standı ile etkileşim halinde olan kullanıcılar

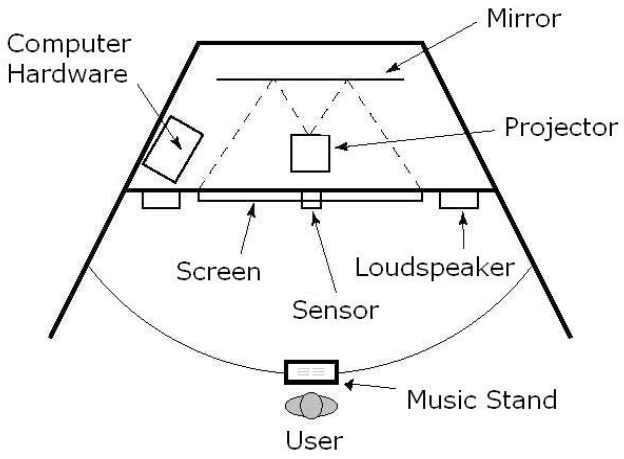
Micon, orkestra idaresinin yapıldığı ve ses ve video ile gerçek zamanlı olarak etkileşime geçilen bir müzik standıdır. Yüksek derecede gerçekçilik için open-GL tabanlı render teknolojisi ile çalışan gösterge sayfalarına sahiptir. Bu sayfalar, kullanıcıya müzik ile ilgili bilgiler vardır. Kullanıcının uzmanlık derecesine göre belirlenmiş karmaşıklık düzeylerine sahip üç farklı gösterge sayfası modeli vardır. Gerçek zamanlı olarak çalışan hareketli görüntü çağırma sistemi ile kullanıcının orkestrayı daha iyi yönetmesi sağlanmaya çalışılmıştır. Bu sistem farklı kullanıcı seviyeleri için farklı tepkilerde bulunur. (Şekil 1.12)



(Borchers ve diğ., 2006)

Şekil 1.12 Micon sisteminin kullanıcıya için sunduğu gösterge sayfası

Micon sistemi, müzikal sonuçları kullanıcıya göstermek ve otomatik olarak birbirini takip eden müzik örneklerini kullanıcı kontrolleri ve kararlarına göre yönlendirmek üzerine kuruludur. Bunun için, bir perde, kullanıcı hareketlerini algılayan sensörler, perde arkasından kullanıcının görüş alanına yansıtılan projeksiyon ve hoparlör sisteminden oluşur. Kullanıcı davranışlarını algılayan sensörler ile alınan görüntüler, bilgisayar yazılımları ile anlamlı verilere dönüştürülür ve buradan çıkartılan sonuçlarla müziğin nasıl ilerleyeceğine karar verilir. Verilen karar doğrultusunda da hoparlörlerden çıkan müzik ve perdeye yansıtılan görüntü belirlenir. (Şekil 1.13)



(Borchers ve diğ., 2006)
Şekil 1.13 Micon aracının sahip olduğu düzenek

Micon çalışması, dijital teknolojilerin gelişmesi ile birlikte etkileşimli ses sistemlerinde kullanıcı kontrollerinin hangi düzeye geldiğini göstermesi açısından önemlidir. Kullanıcı hareketlerinin sese dönüştürüldüğü bu çalışmada, kullanıcı için ses dönüşünü gösteren ve kullanıcının geri beslenmesini sağlayan bir arayüzün gerçek zamanlı olarak çalışıyor oluşu da bizim için önemli bir noktadır.

Kullanıcı hareketlerinin müziğin ve sesin ilerleyişini belirlediği bir düzenekten, 2007 yılında yapılan bir çalışma ile, bu davranışların direk olarak enstrüman olduğu bir düzeneğe geçilmiştir (Bell ve diğ., 2007). MMMS (The Multimodal Music Stand) adlı bu çalışma, kullanıcı olarak müzik icracılarını hedef belirler. Amaç, sınırsız performans hareketleri ile etkileşimli müzik kontrolü sağlamaktır. Duyarlı elektronik alanlar, ses

analizi, ve bilgisayar grafikleri kullanılarak, bir performansçının vücut hareketleri yakalanır ve gerçek zamanlı çalışan bir yazılım ile bu yakalanan hareketlere göre başarılı bir şekilde müzikal performans içerisinde sessel çağrılar da bulunulur. Micon geniş bir kullanıcı yelpazesini amaçlayan bir etkileşimli müzik enstalasyonu iken, MMMS profesyonel kullanıcılar için tasarlanmış bir etkileşimli performans aletidir.

MMMS, herhangi bir performansçının geleneksel tekniğini, enstrümanını fiziksel olarak değiştirmeden bunun sessel imkanlarını genişletmeye yönelik bir çalışmadır. Böylece etkileşimli müzik performanslarında sıradanlığı kırıp, farklı ve deneysel sonuçlar elde etmeye çalışılmıştır. Birçok performansçı, enstrümanlarına fiziksel müdahale de bulunulmasına asla izin vermemektedir. Ancak MMMS bu konuda herhangi bir kısıtlama getirmeden çözüm önermektedir. Özellikle mikrofonlar, kameralar ve elektronik sensörler ile performançının hareketleri yakalanır. Böylece direk enstrüman yerine performansçının hareketleri değerlendirilir. Bu durum, performansçıya hiçbir kısıtlama getirmez. Elbette bu yeni önerilerin kalıcı olabilmeleri için, kullanılabilir ve uzman kontrollerine imkan tanıyan bir yapıya sahip olması gerekir. Bu şekilde kendine has bir repertuar oluşturabilecektir.

MMMS tasarlanırken ilk olarak şu hedefler belirlenmiştir:

- Performansçıya herhangi bir kısıtlama getirmeyen etkileşimli bir elektro-akustik performans imkanı sağlamak
- Enstrümanlarda hiçbir fiziksel değişikliğe gitmeden enstrümanın performans imkanlarını genişletmek
- Performansçının hareketlerini yakalayarak, bunları uygun ses bireşim ve dönüşümlerine tabi tutmak
- Çoklu-model analiz sistemleri kullanarak, çağrı keşiflerinin imkanlarını arttırmak

Böylece, sadece tek bir enstrüman için hazırlanmış bir arayüz değil, farklı bağlamlarda kullanılabilen ve daha önceden tanımlanmış hareketleri kimliklendirip bunları gerçek zamanlı ses bireşimlerine dönüştüren bir müzik standı gerçekleştirilmiş olur. (Şekil 1.14)

(Kaynak: Bell ve diğ., 2007)

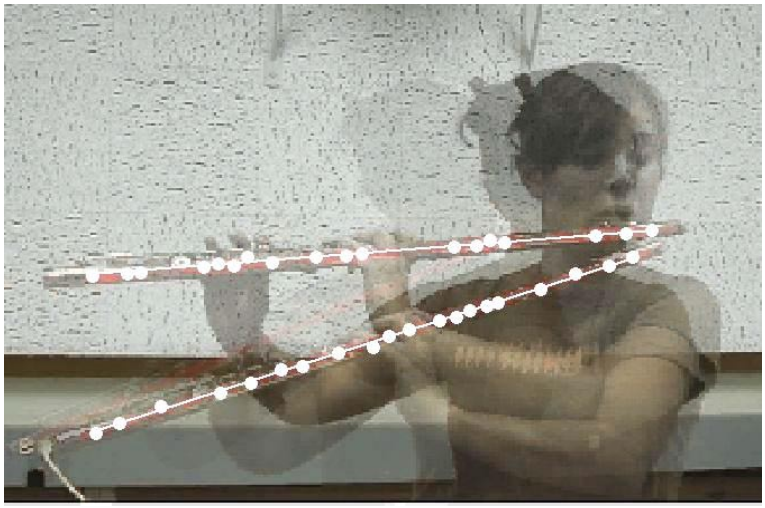
Şekil 1.14 MMMS sisteminin elektronik algı alanını oluşturan antenler ve mikrofonlar

MMMS, etkileşim öğeleri olarak temelde, performansçının yaptığı direk müzik üretimi ile ilgili olmayan yardımcı hareketlerle ilgilenmektedir. Performansçının vücut hareketleri, bu etkileşimli müzik sistemi kaynak olarak kullanılarak, seyircinin esas olarak müziğe katılmayan bu unsurları da duyması sağlanmaktadır.

MMS üç parçadan meydana gelmektedir:

1. The Multimodal Input Analysis
2. Multimodal Detection Layer
3. Audio Synthesis and Transformation

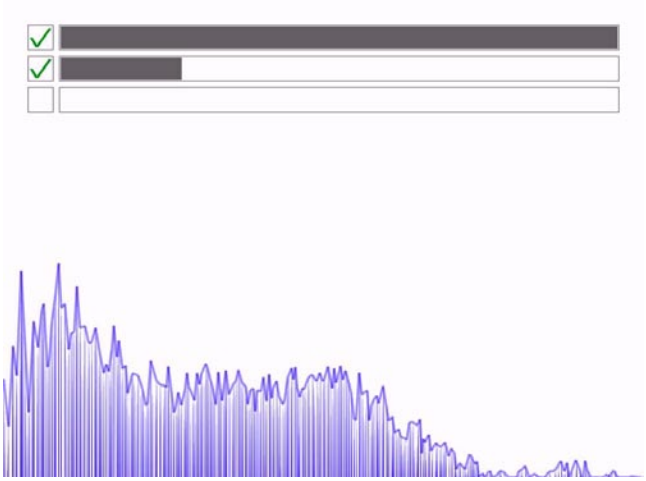
The Multimodal Input Analysis parçası, elektronik duyarlı alan duyusunu ve görsel, duysal analizi kapsar. (Şekil 1.15)



(Kaynak: Bell ve diğ., 2007)

Şekil 1.15 The Multimodal Input Analysis kısmında performansçının hareketlerinin yakalanması

Multimodal Detection Layer, giriş verilerini analiz eder ve uygun olan kullanıcı tanımlı koşulları tetiklemek için çağrı gönderir. (Şekil 1.16)



(Kaynak: Bell ve diğ., 2007)

Şekil 1.16 Multimodal Detection Layer kısmında verilerden anlamlı sonuçların çıkarılması ve kullanıcı kontrolü

Audio Synthesis and Transformation kısmı ise, tetiklemeler ile ilgili sürekli dinleme yapar ve ilgili tetikleme geldiğinde gerekli etkiyi üretir. Bunlar, geleneksel performansçıya, ek müzikal kontrol boyutları kazandırmaktadır. Direk olarak etkileşimli müzik bakış açısına nüfus ederek enstrümanları ve kendilerini teller ile kısıtlamadan yeni etkileşimleri imkanlı hale gelir.

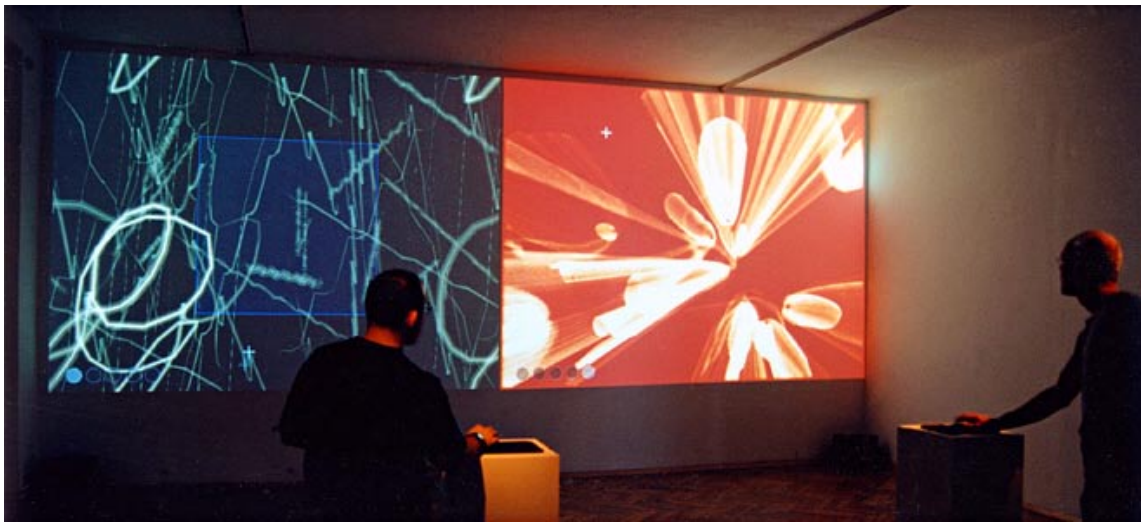
MMMS'de kullanılan duyarlı elektronik alan teknikleri, çalışmamızda daha önce anlattığımız Theremin'in devre topolojisinin dijital ağırlıklı bir yorumuna dayanır. Ek olarak, kamera ve mikrofonların da olması, farklı çağrı kontrollerinin oluşmasını sağlar. Ayrıca, sistemin geri beslemelerinin ve sessel sonuçlarının neler olduğunun görülmesi ve bunlara ait deneyimlerin geliştirilmesi için provalar yapılması gerekmektedir. Bu provalar yanlış tetiklemeleri en aza indirmek için gereklidir.

Burada bizim için önemli olan noktaların başında, direk ses ile ilgili olmayan bir ortamdan ses verilerin oluşturulması ve dönüştürülmesi vardır. Farklı imkanların değerlendirilmesi ve yaratılması için, kullanıcının bunlar ile istediği gibi ayarlama yapabilme imkanının olması önemlidir. Performansçı çalışmalarını daha ileri noktaya götürebilmek için gerekli provaları yapmak durumundadır. Bu noktalar bizim çalışmamız için oldukça önemlidir. Bir diğer önemli alan, MMMS sisteminin üzerine oturduğu modüllerdir. Sistem, verileri alma ve analiz etme, bu verilerden anlamlı sonuçlar çıkarma ve bu sonuçlardan gerekli sesleri üretme işlemlerini yerine getiren üç modülden oluşmuştur. Bu modüllerin başarı ile yan yana gelmesi bizim çalışmamız için önem arz etmektedir.

Diğer yandan, etkileşimli bir ortamda görsel ve duysal alanların bir arada bulunduğu ve bu alanların oluşan estetik ürünü birlikte belirledikleri, birçok görsel-işitsel performans araçları yapılmıştır. Bunlardan biri, 2000 yılında gerçekleştirilen AVES (An Audiovisual Environment Suite) adlı çalışmadır (Levin, 2000). AVES, soyut animasyonlar ve sentetik sesler üretmeye yarayan ve gerçek zamanlı olarak çalışan beş adet etkileşimli sistemden oluşan bir takımdır. Bu beş ortamdan her biri, kullanıcıya deneysel bir performans imkanı sunmaktadır. AVES'i oluşturan parçalardan her biri öğrenmesi kolay ve esnek bir kullanım arayüzü ile tasarlanmıştır. Aynı zamanda oldukça özgün ve geniş bir yelpazede performans sergileme imkanları tanıyarak kişiselleştirilmiş anlatımlar oluşturulmasını sağlamaktadır.

Bu parçalar, sürekli olarak devam eden parçacıklardan oluşmaktadır. Bu parçacıkların durumu, tamamen bağımsız ve özgür bir ortamda kullanıcının mouse ile oluşturduğu

hareketlere göre belirlenmektedir. Daha çok soyut ve hareketli olan bir görsel dilin oluşturulması sağlanmaktadır. Aynı zamanda alt seviye ses birleştirmeleri ile oldukça artistik bir form kazanmaktadır. AVES sistemi; sanatın, tasarımın, enstrüman ve araç mühendisliğinin birleştiği bir noktada yer almaktadır. (Şekil 1.17)



(Kaynak: http://acg.media.mit.edu/people/golan/aves/)
Şekil 1.17 AVES sistemi ile etkileşim içerisinde olan kullanıcılar

AVES'i oluşturan beş parça; Yellowtail, Aurora, Floo, Warbo ve Loom adlı çalışmalardır. Bu beş parça sadece bir mouse aracı ile kullanıcılara deneysel performanslar oluşturma imkanı tanımaktadır. Yellowtail, kullanıcının çizgilerinden yola çıkarak, çizgi şekli ve hareket niteliğine göre sesler üretilmesini sağlar. Aurora, kullanıcıya, devingen bir renk bulutu yaratma imkanı tanır. Bu bulutun form değişiklikleri seslerin oluşumunu sağlar. Floo, yumuşak uçlu sarmaşıklar oluşturulmasını sağlar. Bu sarmaşıkların ararlarındaki boşluklar seslerin karakterlerini oluşturur. Warbo, kullanıcıya hareketli kabarcıklardan kompozisyon oluşturma imkanı tanır. İkinci bir mouse işaretçisi yardımıyla bu yapı içerisinde ses tonlarının oluşturulmasını sağlar. Loom, kullanıcıların oluşturduğu işaretler ile çalışır. Kullanıcı ekranda yarattığı çizgiler ile direk olarak seslerin oluşturulmasını sağlar [6].

Bu çalışmalarda özellikle öne çıkan şey, kullanıcıların farklı görsel-işitsel ürünler ortaya koymasının sağlanmasıdır. Ancak bu mekanizma içerisinde, kullanıcı her iki ortamı da aynı anda belirlemektedir. Burada kaynak kullanıcıların oluşturduğu hareketlerdir.

Kullanıcı görsellerden ses üretme tecrübesinden çok, yarattığı hareketler ile her iki ortamında kaynağını oluşturma tecrübesi edinmektedir. Sonuç üründe, görselliğin kendisi daha öne çıkmakta ve ses performansı kullanıcı için sadece bir destek ortamına dönüşmektedir.

Bu alanda yapılan çalışmalar içerisinde, G. Levin'in yürütücülüğünü yaptığı 2001 tarihli bir çalışma, ses kaynaklarının çeşitliliğinin gösterilmesi için önemlidir. Dialtones (A Telesymphony) adlı bu çalışma, seyircilerin cep telefonlarının melodilerini kaynak olarak kullanmaktadır [7]. Her yerde ve her zaman karşımıza çıkan cep telefonlarını bir müzik aygıtı olarak kullanan bu çalışma ses enstrümanlarına bakış açısı bağlamında önemli bir değişime işaret etmektedir.

Dialtones, seslerinin dikkatle oluşturulmuş bir düzen içerisinde tamamıyla seyircilerin mobil telefonlarına yüklenmiş olan melodilerden oluştuğu, ışık ve görüntü teknolojilerinin de kullanıldığı geniş boyutlu bir senfonidir.

Seyirci koltuklarında bulunan katılımcılar, öncelikle kendilerini girişte kayıt ettirirler. Kayıt sonrasında katılımcıların telefonlarına yeni melodiler yüklenir. Katılımcıların kesin konumları ve telefonlarındaki melodiler bilindiğinden dolayı her türlü düzenleme imkanlı hale gelir. Dialtones çalışması; ses, kamu alanı, elektro manyetik alan, iletişim ağları ile ilgili alanlarda oldukça özgün bir yorum getirmiş ve bunu bir müzik senfonisine dönüştürmüştür. (Şekil 1.18)

(Kaynak: http://www.flong.com/projects/telesymphony)

Şekil 1.18 Dialtones çalışmasının katılımcıları ve performansçıları

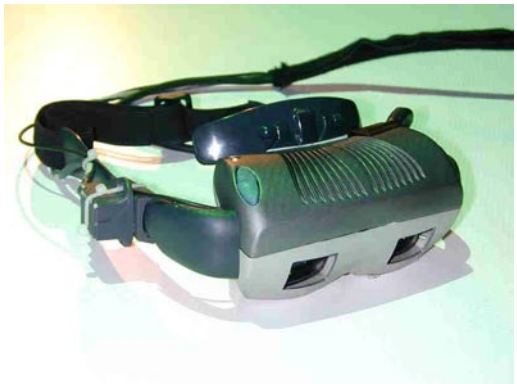
Bir görsel müzik enstrümanı olarak tasarlanan yazılım ile oluşturulan bu performans, yaklaşık onar dakika süren üç ayrı kompozisyondan oluşur. Bu parçalar sırasıyla, sadece katılımcılarda bulunan telefonların melodileri ile yapılan bir performans, solo yapılan ve on adet telefonun farklı yollarla ses aracına dönüştürüldüğü bir performans ve grup halinde yapılan bir performanstan oluşmaktadır. Bu üç performans, bu fikrin ne kadar farklı ve zengin sonuçlar doğurabileceğini gösterebilmek için gerçekleştirilmiştir.

Bu çalışma ses kaynağının özgün kullanımı dolayısı ile önem arz etmektedir. Aynı zamanda müziğin mekan içerisine yayılması uzay alanının da aktif olarak müziğe katılmasını sağlamıştır. Ancak bu mekanizmanın içerisinde, katılımcıların sağladığı etkileşim oldukça yoğun gibi görünse de, sürecin ilerleyişinde performansçıyla kurulan hiçbir diyalog anı yoktur. Katılımcılar sadece, cep telefonları ile orada bulunmaktadırlar. Ancak bundan sonra hiçbir konuda kontrolleri olmamaktadır. Buna ek olarak, görsel müzik enstrümanı olarak tasarlanan bu yazılım, görselliği sisteme destek haline indirgemiş görünmektedir.

Levin G. ve Lieberman Z. Tarafından ses ve görüntü arasında oluşturulan etkileşim çalışmaları arasında 2002 ve 2003 yazları arasında yapılan üç çalışma, bu etkileşimi sesten görüntüye gidecek şekilde genişletmiştir (Levin ve diğ., 2004). Daha önce AVES adlı çalışmada incelediğimiz görsel tavrı amaç edinen, ancak bu çalışmada sesten yola

çıkarak görselliği oluşturmaya çalışan bir yaklaşımla karşı karşıyayız. Her ne kadar görselliği amaç edinen bir çalışma olsa da, ses ile görüntü arasındaki bu etkileşim imkanları ve bunların hangi parametreler aracılığıyla gerçekleştirildiği bizim için önem arz etmektedir.

Özellikle *"sesler bizlere görünseydi, nasıl görünürlerdi?"* gibi bir soru etrafında dolaşarak oluşturulan In-Situ adlı çalışmayı oluşturan üç alt çalışma sırasıyla, Hidden Worlds, RE'MARK ve Messa Di Voce adlarına sahiptir. Bu çalışmalarda, stereographic 3D gözlükler, elektromanyetik konum belirleyen sensörlar, bilgisayar tabanlı görüntüler ve projeksiyon sistemleri kullanılmıştır. (Şekil 1.19)

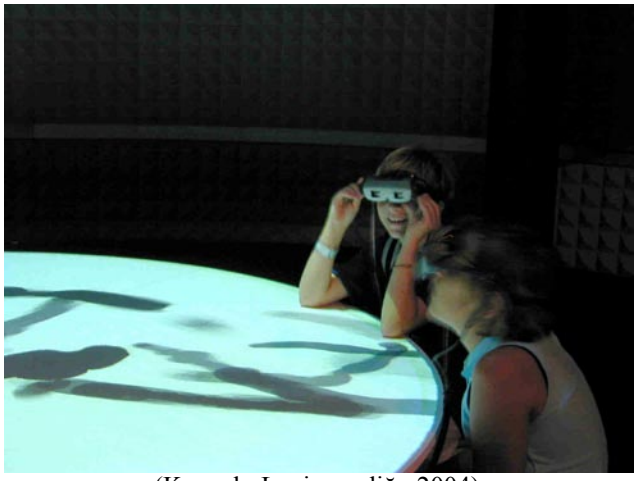


(Kaynak: Levin ve diğ., 2004)
Şekil 1.19 Hidden Worlds çalışmasında kullanılan gözlükler

Bu çalışmalardan ilki olan Hidden Worlds, etkileşimli bir ses ve görüntü enstalasyonudur. Amaç, gerçeklik algısı arttırılmış olan görsel konuşma ortamı yaratmaktır. Bu enstalasyonda, kullanıcılar, stereographic 3 boyutlu gözlükler takarak gerçekçi bir dünyanın içerisine girerler ve özel olarak oluşturulmuş bir masanın etrafında toplanırlar. Masada bulunanlardan biri konuştuğu zaman, bu gözlüklerden bakan kişi konuşan kişinin ağzından çıkan renkli soyut şekilleri görür. (Şekil 1.20) Bu gözlüklerden bakmayan bir kişi ise, alttan projeksiyon ile aydınlatılan masanın üzerinde gözlükle görülen şekillerin iz düşümü olan gölgeleri görecektir. (Şekil 1.21)

(Kaynak: Levin ve diğ., 2004)

Şekil 1.20 Özel olarak tasarlanmış 3 boyutlu gözlükten bakıldığında görülen görüntü



(Kaynak: Levin ve diğ., 2004)

Şekil 1.21 Gözlükle bakmayan katılımcıların görebildiği masa üzerindeki gölgeler

Oluşturulan bu görüntüler, sesin yüksekliği, tınısı ve tonu göz önüne alınarak oluşturulur. Böylece sesin görsellik kazanarak, konuşma edimini farklı bir boyuta geçirmektedir.

Sesin görselleşmesi konusu çerçevesinde yapılan ikinci çalışma olan RE'MARK adlı sistem, Hidden Worlds çalışmasından farklı olarak, mikrofonla aldığı katılımcının seslerini ayrıştırır ve bilgisayar grafikleri ile bunlara uygun soyut şekiller ya da harf siluetleri oluşturur. Bu grafikler, kullanıcının kafasının gölgesinden başlayacak şekilde oluşturulur. Video projeksiyon ile kullanıcının gölgesi ve oluşturulan şekiller aynı ekrana yansıtılır.

Eğer kullanıcının sesi belli bir harf karakteri ile eşleşemez ise soyut şekiller, eğer bir harf karakteri ile eşleşebilir ise ilgili harf karakteri ekrana yansıtılır. (Şekil 1.22)

(Kaynak: Levin ve diğ., 2004)

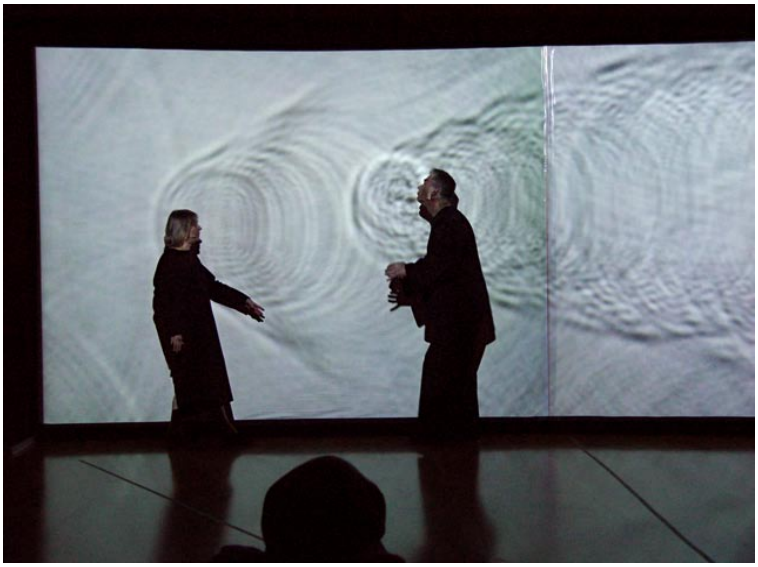
Şekil 1.22 RE'MARK sistemi ile etkileşim halindeki kullanıcılar

Sesin görselleştirmesi için yapılan In-Situ serisinin son halkasını oluşturan Messa Di Voce adlı çalışma, diğer iki çalışma gibi her kesimden kullanıcıya değil, sahne performanslarında bulunacak olan profesyonel vokalistlere yönelik bir konser performansıdır. Bu sistemde vokalistlerin konuşmaları, bağrışları ve şarkıları ile yaratılan gerçek zamanlı görseller üreten etkileşimli bir yazılım desteği vardır. Vokalistlerin oluşturduğu uzmanlık içeren ince nüanslara cevaplar veren bir sistemdir. Soyut şekiller ve sentetik görüntüler üreten bu sistem, kullanıcıdan gelen sesleri ayrıştırarak yüksek ifade gücüne sahip grafikler üretir. (Şekil 1.23)

(Kaynak: Levin ve diğ., 2004)

Şekil 1.23 Messa Di Voce sistemi ile etkileşim halinde olan performansçılar

Bu sistemin temelinde, gerçek zamanlı olarak çalışan görüntü ve ses algoritmaları vardır. Kamera ile performansçının kafasının lokasyonu alınır. Bu lokasyonlar performansçının mikrofon ile alınan sesinin analizi sonucunda oluşturulan grafiklerin perdeye nasıl yansıtılacağını belirlemek için kullanılır. Sonuç olarak, sistem, grafikleri performansçının arkasına yansıtarak ses ve şarkılar ile grafiklerin sıkı bir ilişkiye girmesini sağlar. (Şekil 1.24)



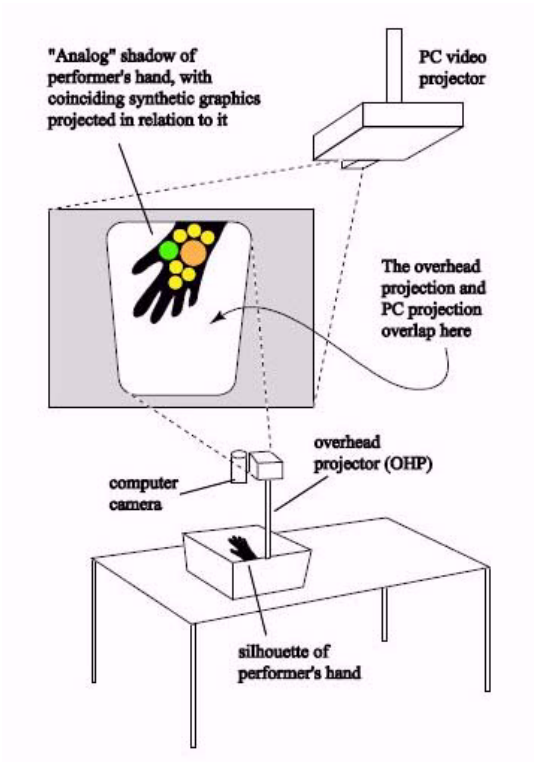
(Kaynak: Levin ve diğ., 2004)

Şekil 1.24 Performansçıların konumlarına göre grafiklerin yerlerinin belirlenmesi

In-Situ serisi içerisinde, en karmaşık mekanizmaya sahip olan Messa Di Voce çalışması, performansçılara daha geniş ifade imkanları sağlayabilmek için, aynı mekanizma temelinde bir çok grafik ve ses analizi algoritması varyasyonuna sahiptir [8]. Bu çalışmaların bizim için önemli olmasının nedeni, görüntü üreten ve ses ayrıştırmaları yapan sistemler olmalarına rağmen, görüntü ve sesin çeşitli şekillerde nasıl etkileşimli hale gelebileceğini göstermelerindendir.

In-Situ serisi tamamlandıktan bir yıl sonra, yine G. Levin ve Z. Lieberman tarafından bu sefer direk olarak görüntüden sese giden The Manual Input Sessions adında bir çalışma yapılmıştır (Levin ve diğ., 2005). Bu yeni çalışmanın özü, In-Situ çalışmasının tersi şeklinde işleyerek, görüntü analizinden sonra seslerin oluşturulmasına dayanır. Böylece, görüntüler sesleri belirleyen unsurlar haline gelirler.

The Manual Input Sessions sistemi, iki projeksiyonun bir araya gelmesi ile oluşmaktadır. Bunlar, geleneksel olan analog tepegöz projeksiyonu ile bilgisayara bağlı dijital projeksiyondur. (Şekil 1.25)



(Kaynak: Levin ve diğ., 2005)

Şekil 1.25 The Manual Input Sessions sisteminin sahip olduğu mekanizma

Bunlara ek olarak, görüntülerin analizine dayanan dinamik sesler ile kullanıcının el siluetlerinin konturlerine cevap veren grafikler üretilir. (Şekil 1.26)

(Kaynak: Levin ve diğ., 2005)

Şekil 1.26 Tepegöz üzerindeki görünüm, bilgisayar grafiklerinin görünümü ve bu iki görüntünün birlikte projekte edilmesi

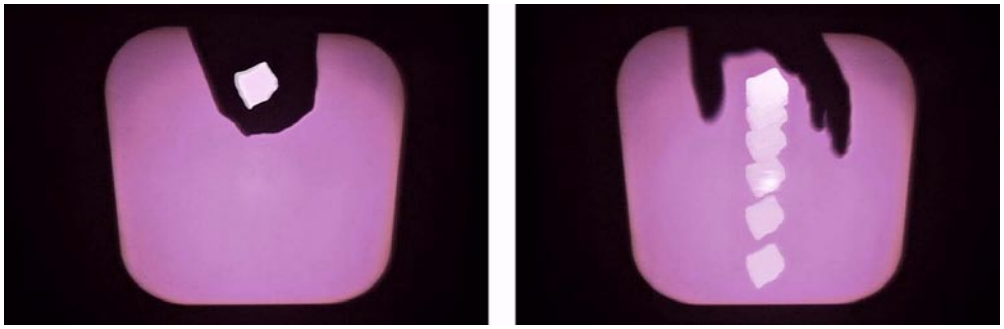
Bu sistemde, asıl etkileşim arayüzünün el siluetlerinden oluşmuş olması bu sistemin öğrenilebilirliğini arttırmaktadır. Aynı zamanda insanların ellerine olan göreceli hakimiyeti bu sistemin zengin ifade gücüne sahip olmasını sağlamaktadır.

Bu araç tasarlanırken öncelikli olarak ele alınan ve bizim için de önem arz eden kriterler şunlardır:

- Kolaylık / zorluk dengesi
- Tekrarlanılabilirlik / sürekli devam edebilme dengesi
- Yaratma, değişime uğratabilme ve yok etme yetileri
- Ses ve görüntü eşliği

Özellikle gerçekçiliği arttırılmış gölge oyunu olarak adlandırılabilecek olan bu düzenek, NegDrop, InnerStamp ve Rotuni adlarında üç farklı enstrümana sahiptir.

Bunlardan ilki olan NegDrop adlı enstrümanda, el ile oluşturulan konturlerin iç kısmı, aşağıya bırakıldığında düşen sanal bir nesneye dönüşmektedir. Bu nesne, düşmeye başlayıp projeksiyonun çerçevesine çarptığında ses çıkmaktadır. (Şekil 1.27)

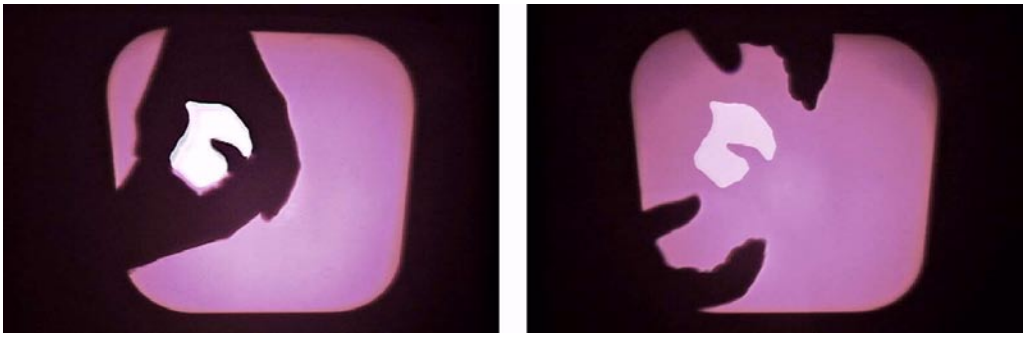


(Kaynak: Levin ve diğ., 2005)

Şekil 1.27 NegDrop enstrümanının çalışma prensibi

NegDrop adlı enstrüman sesleri oluştururken, el siluetinin oluşturduğu şeklin aşağıya doğru düşmesi ile oluşan parametreleri dikkate alır. Kontur alanı, ses frekansını; çarpma hızı, ses yüksekliğini; yatay pozisyon, stereo pan lokasyonunu; kompakt ve noktasal olma durumu, sesin netliğini belirler.

InnerStamp adlı enstrümanda ise, bir öncekinde olduğu gibi yine iç alanlar sanal nesnelere dönüşür. Ancak farklı olarak, bu sanal nesne aşağıya doğru düşmez ve bulunduğu yerde durmaya devam eder. (Şekil 1.28)

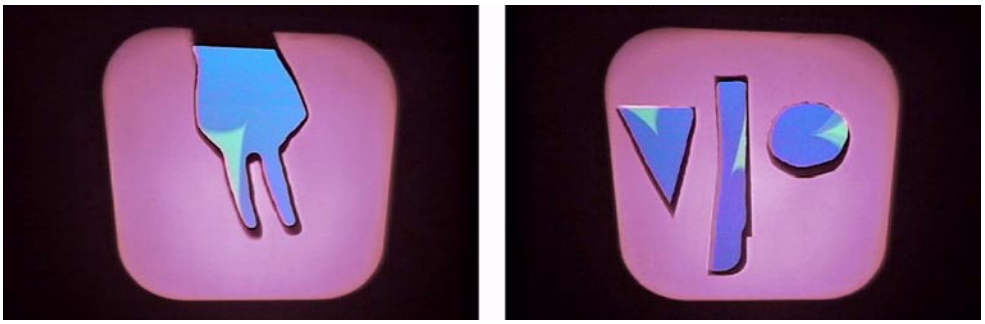


(Kaynak: Levin ve diğ., 2005)

Şekil 1.28 InnerStamp enstrümanının çalışma prensibi

Kullanıcı, bu enstrümanda oluşan sanal nesneyi değişikliğe uğratarak sesleri belirlemeye çalışır. Oluşturulan nesnenin kontur çevresinin uzunluğu, frekansı; yatay pozisyonu, stereo pan lokasyonunu, nesnenin elden kurtulmasından itibaren geçen süre, sesin azalmasını; uzunluğun alana oranı, sesin netliğini belirler.

Rotuni adlı enstrümanda ise, performansçının elinden ya da tepegözün cam tezgahı üzerine konmuş olan herhangi bir şeyin oluşturduğu pozitif kontur alanından ritmik ve melodik tekrarlar oluşturulur. NegDrop ve InnerStamp'ın tersine, Rotuni'de kullanıcı bir iç alan oluşturmak zorunda değildir. (Şekil 1.29)



(Kaynak: Levin ve diğ., 2005)

Şekil 1.29 Rotuni enstrümanının çalışma prensibi

Rotuni adlı enstrüman sesleri oluştururken, herhangi bir pozitif alanın oluşturduğu parametreleri dikkate alır. Kontur çizgilerini oluşturan noktaların merkeze uzaklığı, ses frekansını; yatay pozisyon, stereo pan lokasyonunu; konturun karakteri, sesin netliğini belirler.

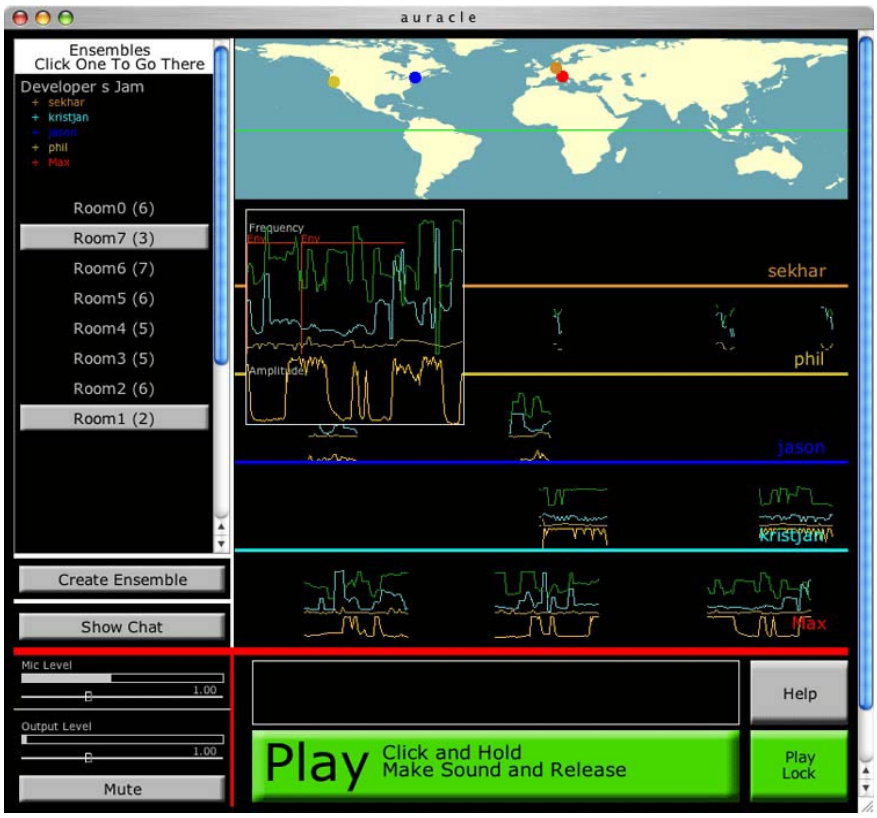
The Manual Input Sessions çalışması, ses kaynağının görüntülerden kaynaklanması ve etkileşim tasarımı olarak elin ve farklı nesnelerin kullanılması ile bizim için önemli özellikler gösteren bir çalışmadır. Diğer yandan, görüntü ve ses arasında birbirine bağlanacak olan özellikler için yapılan seçimler ilgi odaklarımızdan biridir.

Seslerin oluşturulmasının ve belirlenmesinin kontrollerinin yapılabileceği ortamlardan biri de network ortamıdır. Bu konu ile ilgili olarak yapılan bir çalışma Auracle adını almaktadır (Ramakrishnan ve diğ., 2004). Auracle, network üzerinde çalışan ve insan sesi ile kontrol edilen bir grup enstrümanıdır. Web tarayıcıları ile çalışan bu enstrüman, kullanıcıların internet üzerinden gerçek zamanlı olarak ilişkiye girmesi ile ortaklaşa seslerin üretilmesini sağlar. Auracle'ın tasarlanması sırasında, amaçlanan üç kriter yerine getirilmeye çalışılmıştır. Bunlar:

- Etkileşimin insan sesleri ile gerçekleşmesi
- Herkesin birbirinin etkilerini görebildiği bir grup çalışması olması

- Sınırsız bir gelişime açık olmasıdır.

Böylece Auracle, network ve etkileşimli ses kontrolü ile geniş kitleli bir ses üretim ortamı yaratır. (Şekil 1.30)

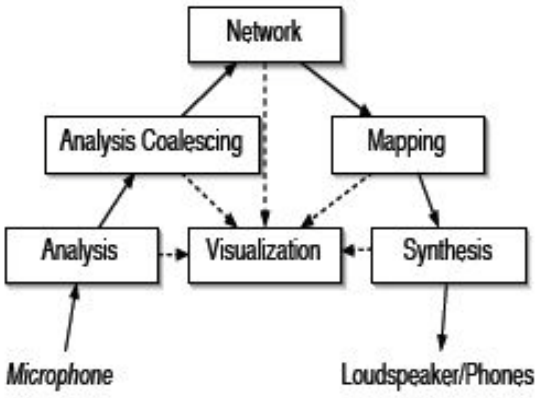


(Kaynak: Ramakrishnan ve diğ., 2004)

Şekil 1.30 Auracle enstrümanının arayüzü

Auracle arayüzü, işlevinin gereği olarak standart bir arayüze sahip değildir. İnternet tarayıcısı çerçeveleri, ikonlar ve klasik menüler bulunmamaktadır. Hızlıca giriş işlemlerini gerçekleştirecek olan butonlara odaklanılır. Yapılan işlemlerin direk sonuçlarını gösteren alan, kullanıcı için bilgilerin geri döndürüldüğü alandır. Auracle sistemi altı adet modülden oluşmaktadır. Öncelikle, mikrofondan alınan sesin analiz edildiği analiz modülü vardır. Buradan gelen veriler daha sonrasında bileşenlerine ayrılır ve sınıflandırılır. Böylece analiz modülünden gelen veriler, anlamlı bilgilere dönüştürülür. Oluşturulmuş olan veriler bu aşamadan sonra network iletişimi modülü ile network üzerinden diğer istemcilere gönderilir. Ardından, sistemdeki her bir istemciden onların verileri ile ilgili bir talepte bulunulur ve bunlar haritalandırma modülü ile bir

biçimde birleştirilir. Böylece ses sürekli olarak kendisini geliştirerek var olmaya devam eder. Birleştirilen bu son bilgiler ses üretim kontrolü olarak kullanılır. Bütün bu aşamaların iz düşümleri, altıncı modül olan görselleştirme modülü içerisinde sürekli olarak görselliğe dökülür. (Şekil 1.31)



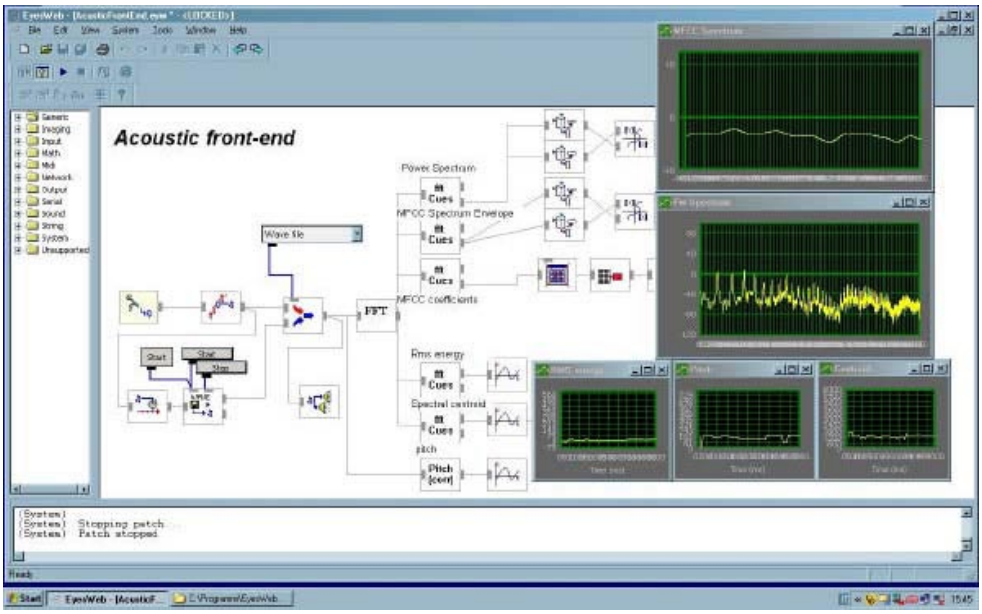
(Kaynak: Ramakrishnan ve diğ., 2004)
Şekil 1.31 Auracle sisteminin çalışma mimarisi

Auracle sistemi Java ortamında programlanmıştır ve kullanıcılar sonuç olarak bir java appletini çalıştırmaktadırlar. Java ortamının, ses kontrolleri ve üretimlerinde başarı ile kullanılması bizim için önemlidir.

Ses ve görüntü işleme konusunda bizim için önemli olan ve incelenmesi gereken çalışmalar arasında özel olarak tasarlanmış ve farklı farklı amaçlar için kullanılabilecek araç geliştirme ortamları vardır. Bu ortamların ortak özelliği, görsel bir arayüz içerisinde, kullanıcının kendi ihtiyaçlarına uygun olan gerekli algoritma ve iş akışlarını ifade eden düzenlemelerin yapılabilmesidir. Bu ortamlarından biri EyesWeb adlı ortamdır (Camurri ve diğ., 2005).

EyesWeb, gerçek zamanlı olarak dans, müzik ve çoklu ortam uygulamalarının tasarlanması ve geliştirilmesi için oluşturulmuş bir görsel arayüz olan etkileşimli bir sistemdir. Açık platform olarak tasarlanmış olan EyesWeb, vücut hareketlerinin işlenmesi gibi kamera kaynaklı veriler ile farklı ortam verilerini işleyerek, farklı ortamlarda çıkışlar vermeye göre tasarlanmıştır. EyesWeb, entegre işlemleri farklı

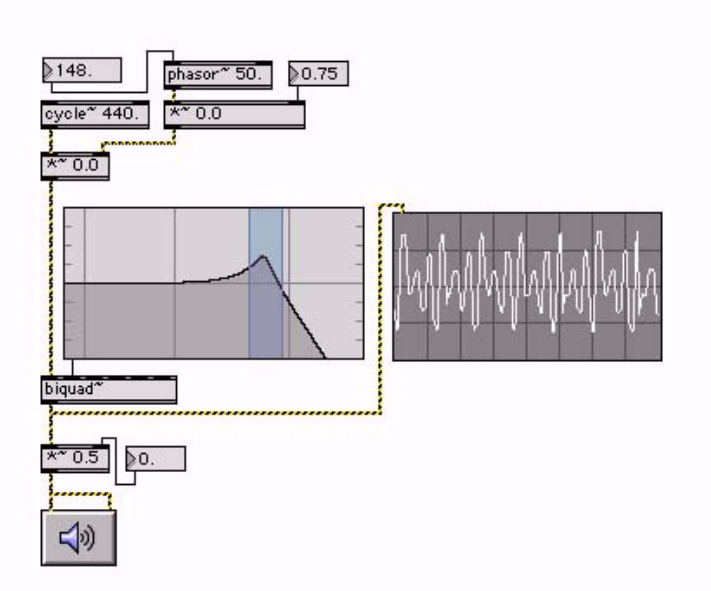
ortamlar için destekleyen bir yapıya sahiptir. Özellikle etkileşimli performansların düzenlenmesi için görsel programlama dili barındırır. Sahip olduğu kütüphaneler içerisinde; kamera, kablosuz vücut sensörleri, ses ve MIDI girişleri, seri bağlantılar, network, klavye ve mouse gibi ortamlardan veri girişi alan kütüphane, matematiksel ve filtre hesaplamalarını yapan kütüphane, imaj işlemlerini yapan kütüphane, ses ve MIDI işlemlerini yapan kütüphane, network bağlarını kuran kütüphane, hareket analizi yapan kütüphane, gerçek zamanlı ses ve görüntü çıkışlarını düzenleyen kütüphane ve görüntü, ses, MIDI ve network çıkış bilgilerini oluşturan kütüphane olmak üzere birçok temel kütüphaneyi barındırır. (Şekil 1.32)



(Kaynak: Camurri ve diğ., 2005)

Şekil 1.32 EyesWeb yazılımının arayüzü

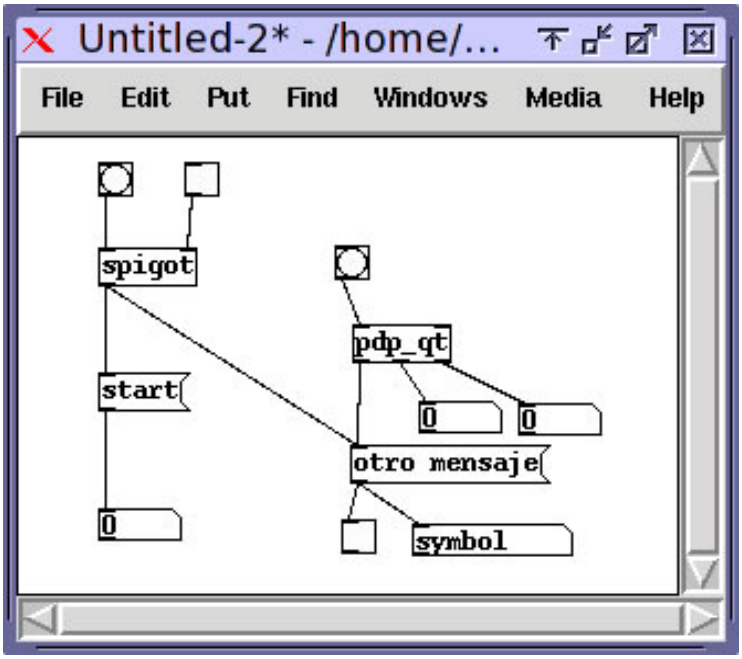
Ses, görüntü, video gibi çoklu ortam elemanları ile ilgili uygulamalar geliştirmek için tasarlanmış olan görsel uygulama geliştirme ortamlarından biri de Max/Msp'dir [9]. Bu görsel program geliştirme ortamı, herkesin çoklu ortama yönelik olarak kendi yazılımını üretebilmesi için, görsel araçlar ve nesneler barındırır. Daha çok ses ve MIDI işlemeleri için geliştirilen bu programda, kullanıcı elemanlar arasında bağlar kurarak çalışır. (Şekil 1.33)



(Kaynak: http://www.cycling74.com/products/maxmsp)
Şekil 1.33 Max/Msp aracının çalışma ortamı

Max/Msp adlı yazılımda da, diğer grafik programlama dillerinde olduğu gibi fonksiyonlar ve objeler arasında bağlantılar kurarak devam eden bir çalışma yapısı vardır.

Bu şekilde çalışan bir başka görsel programlama dili Pure Data adlı yazılımdır [10]. Bu yazılım, diğerleri gibi ses ve MIDI işleyerek etkileşimli bilgisayar müziği geliştirme ortamı sunmaktadır. Pure Data görsel programlama dilinin çalışma şekli ve veri yapısı, C gibi diğer programlama dillerinin çalışma mantığı ve veri yapıları ile benzeşmektedir. Kullanıcı, her türlü veriyi ve işlemleri görsel akış şemalarında görebilir ve bunları düzenleyebilir. Ses, video, grafik gibi farklı ortamların verilerini işleyerek çoklu ortam uygulamalarına imkan veren Pure Data programı, Max/Msp gibi ortamların kullanıcı için oluşturduğu kullanım zorluklarının basitleştirilmesi gibi bir çaba ile tasarlanmaya başlamıştır (Puckette, 1996). (Şekil 1.34)

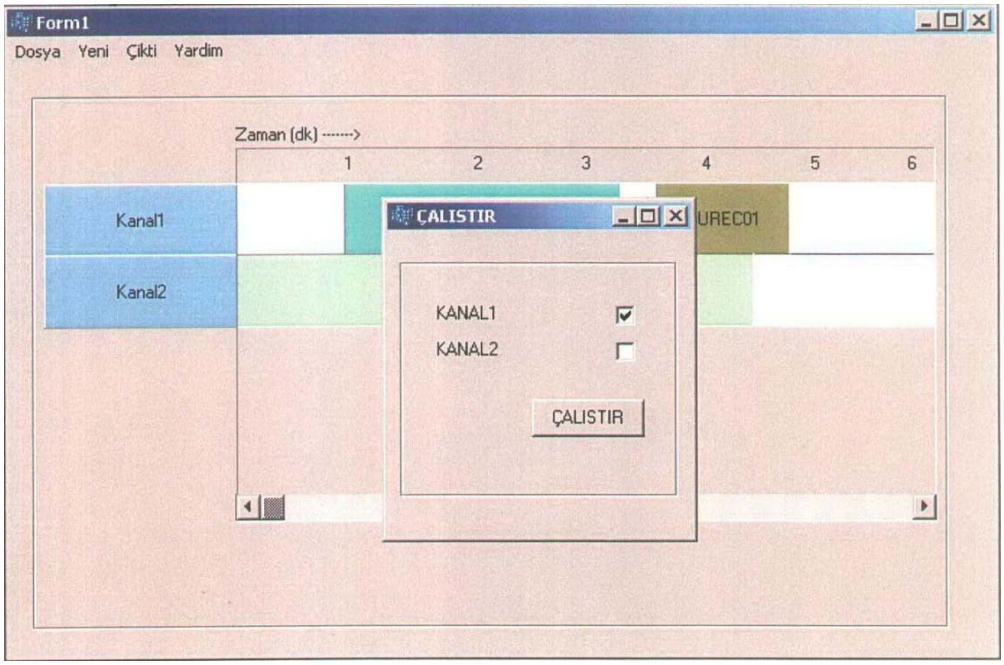


(Kaynak: http://puredata.info/)

Şekil 1.34 Pure Data yazılımının çalışma ortamı

Burada incelemiş olduğumuz EyesWeb, Max/Msp ve Pure Data gibi programlar çoklu ortam uygulamalarını geliştirmek için sunulmuş olan çalışmalardır. Ancak bu uygulama geliştirme ortamları, kendisini kullanacak olan kişilerin uzun süre tecrübe etmelerini gerektirmektedir. Her ne kadar görsel bir arayüz içerisinde kullanıcıların işlerini kolaylaştırabilmek için hazırlanmış olsalar da, amacı sadece ses üretmek olan kullanıcılar için fazlasıyla karmaşık bir yapıya sahiptirler. Bu ortamlar içerisinde başarı ile uygulama geliştirebilmek, özellikle iş akışı ve veri yapıları hakkında belli bir düzeyde bilgi sahibi olmayı gerektirmektedir. Bu durum, farklı giriş ve çıkış verilerini işleyebilseler de, çoklu ortam alanında belli bir amaca yönelik olan ve tecrübesiz kullanıcıların (özellikle performans icracıların) başarı ile kullanabileceği yazılımların geliştirilmesi ihtiyacını ortadan kaldırmaya yetmemektedir.

Performans icracılarının yukarıda anlatılan sorunlarla karşılaşmaları, kendi alanlarında kullanılmaya yönelik olarak özelleşmiş yazılımların geliştirilmesini gerekli kılmaktadır. Bu konuda yapılan çalışmalardan biri, modern dans koreograflarının kullanımı için tasarlanmış olan bir araçtır (Kuşçu ve diğ., 2005). Bu araç, sahne performanslarında kullanılan etkileşimli sistemlerin kontrollerini koreografların belirleyebilmesi için, bir ses ve görüntü kurgu programı arayüzü sunmaktadır. (Şekil 1.35)



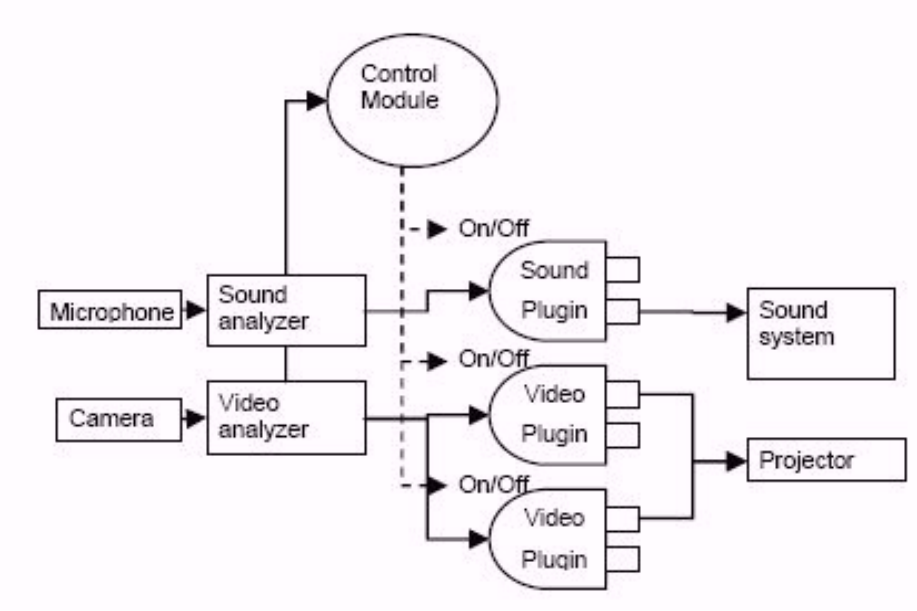
(Kaynak: Kuşçu ve diğ., 2005)

Şekil 1.35 Modern dans koreograflarının etkileşim kontrollerini yönetebilmesi için tasarlanmış olan kurgu programının arayüzü

Böylece koreograflar, sadece programcılar tarafından kendilerine sunulan etkileşim sınırlarında değil, kendi kontrolleri ile belirlenen parametrelerin dahilinde tasarımlar gerçekleştirebileceklerdir. Kullanıcı, alınan verilerin yorumlanması ile oluşan durumlar karşısında hangi kararların alınacağını ve bu verilerin sistemde hangi parametreleri etkileyeceğini kompozisyon sürecinde belirleyecektir. Bu amaç için tasarlanan aracın sahip olması gereken nitelikler şu şekilde belirlenmiştir:

- Birden çok ortama girdisi olmalı
- Birden çok ortama çıkışı olmalı
- Etkileşim seçenekleri düzenlenebilmeli
- Genişlemeye açık olmalı
- Zamanlama sistemi olmalı
- Tetikleme sistemi olmalı
- İş yükü dağıtabilmeli
- Kullanıcı dostu olmalı

Farklı ortam verilerini değerlendirip yine farklı ortamlarda çıktılar hazırlayan bu aracın sistem yapısı modüler bir yapıya sahiptir. Bunlar; giriş işlemcilerinden gelen bilgiyi yorumlayarak ve zamanı izleyerek hangi çıkış modülünün çalışacağına karar veren kontrol modülü, giriş donanımından gelen veriyi işleyen giriş modülü ve bir veya daha fazla giriş işlemcisinden aldığı veriyi kendi yöntemleriyle işleyerek çıkış donanımına parametre olarak veren çıkış modülleridir.

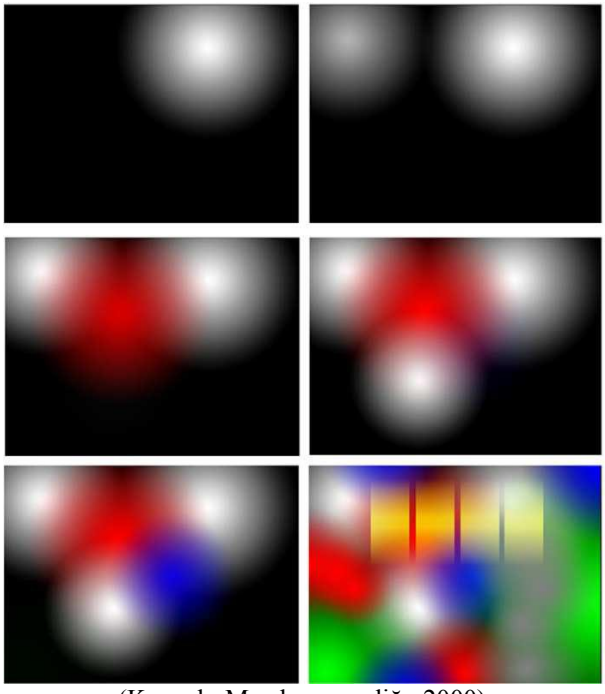


(Kaynak: Kuşçu ve diğ., 2005)
Şekil 1.36 Sistemin modüler yapısı

Her ne kadar ses kullanımı için özelleşmemiş olsa da, kullanıcıları herhangi bir programlama bilgisi olmayan modern dans koreografları olan bu aracın temel kaygısı bizim için önemli bir özelliktir. Programlama konusunda bir bilgisi olmadığını varsaydığımız işitsel-sanat icracılarının kullanımı için tasarladığımız aracın teknik ayrıntılardan arındırılmış olan bir arayüze ihtiyaç duyması gerektiği düşünüldüğünde, koreograflar için yapılan bu çalışma gibi kullanıcı dostu bir yapıya sahip olması gerektiği görülmektedir. Bu çalışmanın bir diğer önemli özelliği, sistem yapısı olarak modüler bir çözümlemeye sahip olmasıdır.

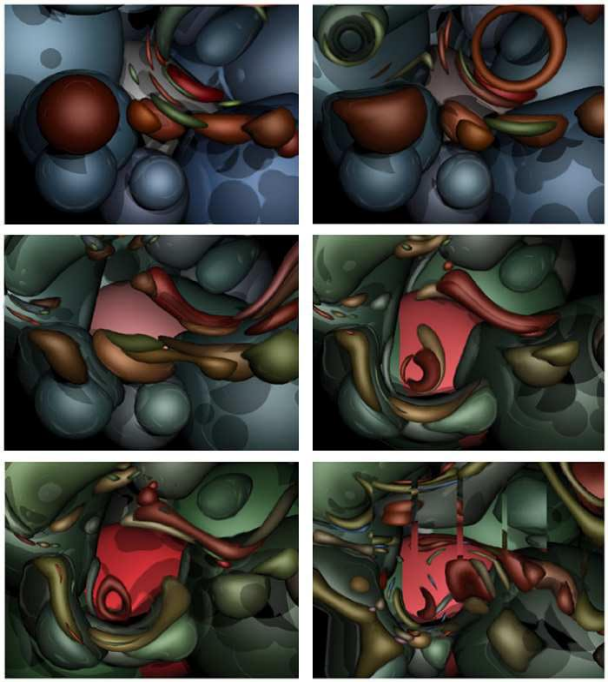
Direk olarak ses ortamı ile ilgili olmayan ancak bizim çalışmamıza esin kaynağı olabilecek bir diğer çalışma, The Painting Camera adlı yazılımdır (Meadows ve diğ.,

2000). Bu çalışmada basit ve sezgisel olarak görsel sonuçlar elde etmeye yarayan bir render tekniği geliştirilmiştir. Özellikle fütürist, kübist ve soyut ressamların kullanabileceği bir teknolojidir. Üç boyutlu sanal ortamda bulunan kameraların aldığı görüntülere ve camera-control-image'lere göre çalışan bir resim oluşturma tekniğidir. Bu şekilde birbirinden oldukça farklı olan resimsel sonuçlar elde edilebilmektedir. Kullanıcı kendisinin belirleyebildiği ve kamera görüntüleri için parametre teşkil eden camera-control-image'ler ile istediği soyut etkiyi elde edebilmektedir. The Painting Camera tekniği herhangi bir render motoruna rahatça adapte edilebilen bir araçtır. (Şekil 1.37 ve Şekil 1.38)



(Kaynak: Meadows ve diğ., 2000)

Şekil 1.37 The Painting Camera tekniğinde kullanılan camera-control-image örnekleri



(Kaynak: Meadows ve diğ., 2000)

Şekil 1.38 The Painting Camera tekniği ile oluşturulan soyut resimler

Sanal kamera ile alınan üç boyutlu görüntünün, camera-control-image'lerden alınan parametreler ile dönüşüme uğratılması üzerine yükselen bir sistem olan The Painting Camera tekniği, sanal nesnelerin ya da fotoğrafik nesnelerin camera-control-image'leri olarak kullanılma imkanı ile farklı sonuçlar elde edilmesini imkanlı kılmaktadır. Bu çalışmanın bir sonraki aşaması, farklı kameraların verilerinin harmanlanması ile kübist etkilere sahip görüntülerin oluşturulmasıdır (Smith ve diğ., 2004). Böylece çok kameralı bir sistemle verilerin değerlendirilme zenginliği arttırılmıştır.

The Painting Camera tekniğinin bizim için önem teşkil eden yanı, görüntü oluşturma teknolojisi konusundaki özgün niteliği ya da teknik olarak sahip olduğu alt yapı değildir. Burada bizim için esin kaynağı olarak önem teşkil eden özellik, kameranın (bu çalışmada üç boyutlu ortamdaki sanal kamera) işlevinin değiştirilerek bir şekilde resim aracına dönüştürülebilmesi fikridir. Kameranın farklı bir amaca yönelik olaan bu kullanımı, bizim çalışmamız için ufuk açıcı olmuştur.

1.4 HEDEF

Yukarıda yaptığımız incelemeler ile birlikte, tasarladığımız aracın fikirsel arka planını ve esin kaynağı olan çalışmaları incelemiş olduk. Bu sürecin ardından, çalışmamızın başlığı kamera kaynaklı işitsel-sanat aracı olarak belirlendi. Tasarladığımız araçta hedefimiz, kullanıcılara (işitsel-sanat icracılarına) sunulacak olan bir enstrümanın düzeneğini tasarlamak ve bunun uygulamasını gerçekleştirmektir. Kullanım amacı ve şekli açık ve bütünlüklü olan bir araç tasarlamak temel kaygımızdır. Böylece kullanıcılar giriş ve çıkışları belli olan bir sistem ile karşı karşıya gelecek ve kendilerini bu yapı içerisinde kaybolmuş hissetmeyeceklerdir. Bu özellik, programlama bilgisi yetersiz olan işitsel-sanat icracılarını düşündüğümüzde daha da gerekli olmaktadır.

Kamera kaynaklı bir işitsel-sanat aracı temel olarak, kameradan alınan verilerin gerçek zamanlı olarak işlenmesi ile oluşturulan parametrelerden, seslerin yaratılması için yararlanılmasına dayanır. Burada etkileşimin kaynağını, kameranın kullanımı ile seslerin nitelik ve niceliklerinin belirlenmesi oluşturur. Bunun için, donanım ve yazılım arasında bir bağ kurulmalı ve birlikte çalışabilmeleri sağlanmalıdır. Bu sistemde, kamera bir ses enstrümanına dönüştürülmektedir.

Burada özellikle üzerinde durulması gereken bir nokta, bu çalışmanın amacının kullanıcılara bir ses enstrümanı sunmak olduğundan dolayı, aracın kendi içerisinde kullanıcılara hazır ses yapıları sunmayacak olmasıdır. Farklı ürünlerin ortaya çıkması ve geniş bir yelpazeye oturması için bu gerekli görülmüştür. İşitsel-sanat icracıları, sunduğumuz aracın imkanlarını kullanarak görüntü tabanlı seslerin nitelik ve niceliklerini kendileri belirleyebilecektir. Üretilen sesler en basit formlarıyla frekanslar olacaktır. Bu konuda verdiğimiz karar için şu benzetme yapılabilir: Kullanıcılara, cümle kurmaları için hazır kelimeleri değil, sadece harfleri sunmaktayız. Görüntü araçlarının anlamlı bütünlükler sunmak için en küçük yapı olan pikselleri kullanması gibi, görüntü özelliklerinin ses özellikleri ile eşleşmesi üzerine kurulu olan bu işitsel-sanat aracı da, seslerle oluşturulacak olan bir bütünlüklü sunumun temel taşları olan ses frekanslarını kullanıcıya sunar. Böylece kullanıcılar kendi ses ve görüntü hakkındaki bilgi dağarcıkları dahilinde geniş bir üretim imkanına sahip olacaktır.

Bu amacımıza ulaşabilmek için, yani geniş bir kontrol imkanının, gerçekleşebilmesi için, kullanıcının sadece bu aracı tasarlayanların görüntü ve ses parametrelerini ve oranlarını belirleyişine mahkum olmaması gerekmektedir. Kullanıcı kendi inisiyatifleri ile kontroller sağlayabilmeli ve kontroller üzerinde süreç içerisinde değişiklikler yapabilmelidir. Bunun için, aracın kamera ve hoparlörler dışında kullanıcıya sunduğu bir etkileşimli arayüzün bulunması gerekmektedir. Bu arayüz ile ses ve görüntü arasında kurulan ilişkilerin özellikleri belirlenebilmelidir.

Bu arayüzün en temel özelliği kullanıcı dostu olmasıdır. Direk manipülasyona imkan veren bir düzenek olması ve farklı seçenekleri barındırması, kullanıcının işlemler sırasında kafasını karıştıracak olan hiçbir öğe ile karşılaşmamasını gerektirmektedir. Bu düzeyde, ses ve görüntü nicelikleri için herhangi bir sayısal nicelik arayüzde görüntülenmeyecektir. Kullanıcı genel ve anlık tecrübelerinde ses nitelikleri ile direk olarak karşı karşıya gelerek ve yaptığı düzenlemeleri yeniden ele alabilecektir.

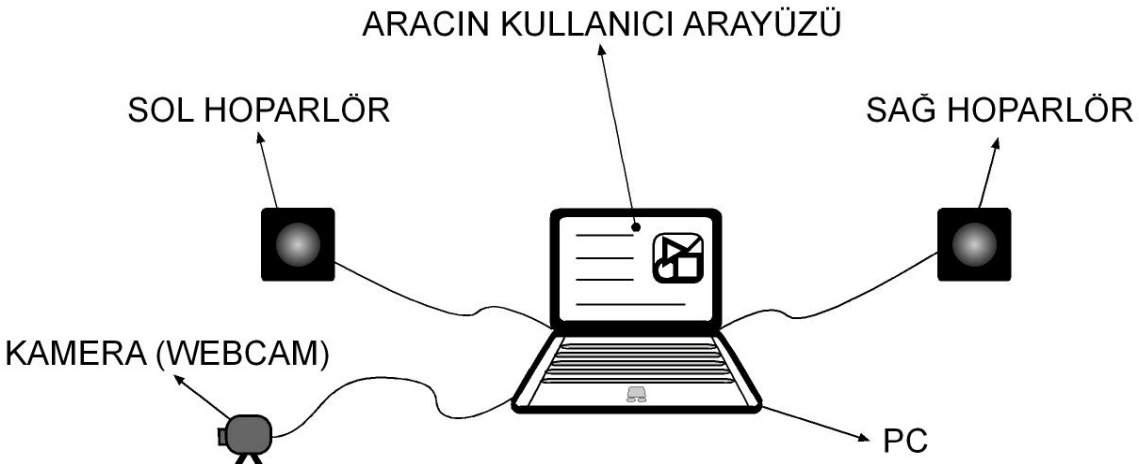
Tasarladığımız sistem, kullanıma yönelik kolaylık konusunda önemli bir dengeyi tutturmak zorundadır. Unutulmamalıdır ki, sunduğumuz araç yeni bir ses enstrümanı önerisidir. Kameranın ve görsel arayüzün ses üretimi için kullanılması sırasında belli bir niteliğin tutturulması ve aracın imkanlarının keşfi için, performans icracılarının belli bir prova ve uzmanlaşma dönemi geçirmesi gerekecektir. Amacımız, bir ses enstrümanı simülasyonu yaratarak kullanıcılara bir müzik deneyimi yaşatmak değildir. Aracın kullanımıyla bütünlüklü bir eser ortaya koymak sanatçının yaratıcılığını ve bakış açısını gerektirmektedir.

Bunun yanında, eğer aracın kullanıcı dostu olması adına direk manipülasyon ve kullanıcı kontrolü özelliklerini geri planda tutarsak, hem ses üretim zenginliği kısıtlanacak hem de kullanıcının başarma hissi azalacaktır. Aynı zamanda eğer aracın kullanımı çok zor olur ise, araç sıkıcı bir etkileşim sunmak riski ile karşı karşıya gelecektir. Bunun için tasarladığımız düzenekte bu dengenin de gözetilmesi önemlidir.

Tasarladığımız araç ile üretilen seslerin çeşitliliği, kullanıcının hayal gücünü kısıtlamamalıdır. Bunun için, kullanıcı tarafından seslerin istendiği değerde ve sayıda yaratılabilmesi, bu seslerin performans sürecinde değiştirilebilmesi, manipüle edilebilmesi ve bu seslerin yine kullanıcı kontrolü ile yok edilebilmesi gerekmektedir. Kullanıcı zamanla kazandığı tecrübeyi tekrarlayarak yaptığı çalışmaların üzerine çıkabilmelidir. Bunun için, aracın kullanılması ile üretilen seslerin tekrarlanabilir olması gerekmektedir.

Bu özellikleri gösteren bir araç, hem teknolojik üretim aşamasında hem de kullanıcının performansı sırasında modüler bir yapı üzerine oturmalıdır. Giriş aygıtlarından alınan verilerin işlenmesi, bu verilerden anlamlı sonuçların üretilmesi, bunların kullanıcı kontrollerine sunulması ve buradan seslerin parametrelerinin oluşturularak çıkış aygıtlarına gönderilmesi süreci programlama sırasında ve aracın çalışması sürecinde modüler bir yapı ile çözülmelidir. Bu durum, kullanıcıya kendiliğinden bir şekilde modüler bir kontrol imkanı sağlayacaktır.

Araç düzeneğinin, maliyeti düşük bir donanım ve yazılım desteği ile gerçekleştirilmesi kararlaştırılmıştır. Böylece, aracın her yerde kullanılabilmesinin ve ulaşılabilmesinin yanında üretimin gerçekleştirilmesinin kolaylaşması da sağlanacaktır. Bu aracın gerçekleştirilmesi için, görüntü verilerinin alınacağı bir kamera ve ses verilerinin çıktı olarak verileceği hoparlörler yeterli olacaktır. Bütün bunların ve kullanıcı arayüzünün üzerinde çalışacağı ortam PC (personal computer) olacaktır. (Şekil 1.39)



Şekil 1.39 Kamera kaynaklı işitsel-sanat aracının mekanizması

2. MALZEME ve YÖNTEM

2.1 MALZEME

Bu kısımda, ‘Giriş’ bölümünde hedef olarak belirlediğimiz kriterlere uygun bir kamera kaynaklı işitsel-sanat aracı ve arayüzü tasarımını gerçekleştirmek için kullanacağımız malzemeleri inceleyeceğiz. Öncelikle tasarımımızın en temel iki ortamından biri olan sesin fiziksel özellikleri incelenecektir. Ardından çalışmamız bilgisayar ortamında gerçekleşeceğinden dolayı, ses teknolojisi açısından analog ve dijital seslerin özellikleri incelenecektir. Ses ortamını ilgilendiren diğer bir teçhizat ise sisteme bağlı olan hoparlörlerdir. Bu nedenle ses malzemesinin temel uzantılarından biri olan hoparlörlerin özellikleri belirlenecektir. Ses ortamı ve buna bağlı olan özellikler incelendikten sonra görüntü ortamının temel belirleyeni olan renklerin fiziksel özellikleri ele alınacaktır. Bundan sonra, görüntüyü belirleyen temel bileşen olan renk, aracımıza kamera cihazı ile katıldığından dolayı kameralar incelenecektir.

2.1.1 Sesin Fiziksel Özellikleri

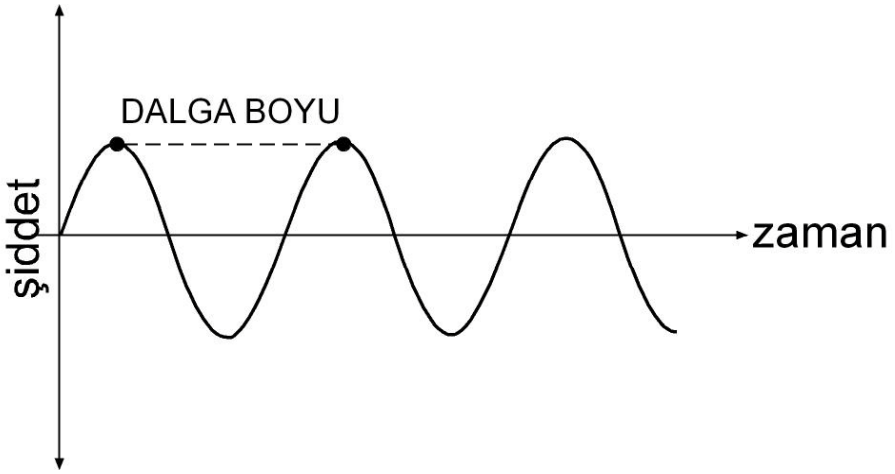
Ses, en genel tanımıyla düzenli ya da düzensiz mekanik titreşimlerin, atmosferik bir ortamda oluşturduğu basınç değişiklikleri ve bunun işitme organında doğurduğu duyumdur. Diğer bir tanımla, akustik bir dalganın doğurduğu işitme duygusuna verilen addır. Fiziksel anlamda dalga oluşumu, maddesel ortamlarda madde aktarımı olmaksızın enerji yayılmasıdır (Sözen, 2003).

Bu durumda, bir ses dalgasının ses olarak algılanmasından bahsedebilmek için, ses kaynağının, iletici ortamın ve bir alıcın olması gerekmektedir. (Şekil 2.1)

Şekil 2.1 Sesin oluşumu için gerekli öğeler ve koşullar

Sesin yayılabilmesi için iletici bir ortama ihtiyaç vardır. Ses dalgaları, ışık dalgaları gibi boşlukta yayılamaz. İletici ortamı oluşturan parçacıklar, ses kaynağına değme yüzeyinde, kaynağın yaptığı titreşime uyarak titreşmeye başlarlar. Titreşmeye başlayan parçacıklar, kendi etraflarındaki diğer parçacıkları harekete geçirir. Ortam içerisinde oluşan bu harekete ses dalgası denir. Bu dalga kendini yineleyerek sürüp giden bir harekete sahiptir. Bu yüzden bunlara "periyotlu hareket" denir.

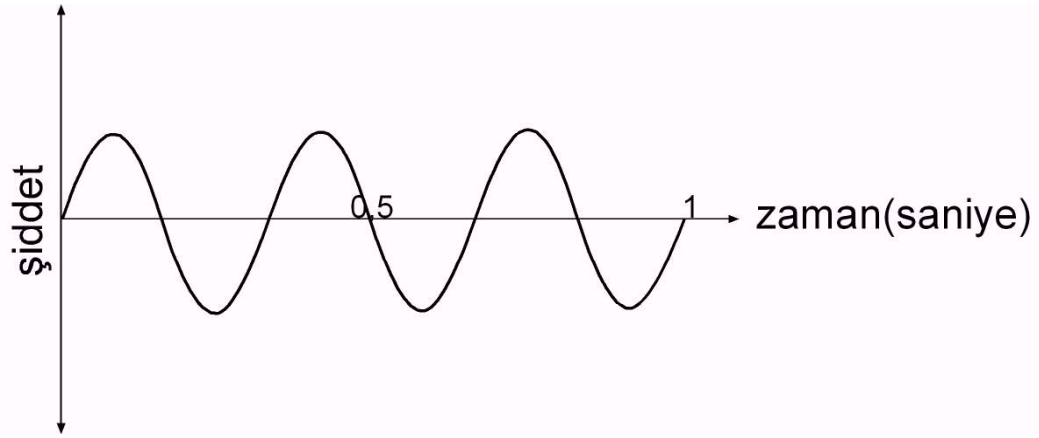
Bu titreşim dalgalarının tanımlanmasında başlıca dört parametre vardır. Bunlar; dalga boyu, dalga hızı, dalga frekansı ve genliktir. Dalga boyu, dalga üzerinde özdeş olarak davranan herhangi iki nokta arasındaki minimum mesafedir. Dalga hızı, dalgaların içinde oluştukları ortamın özelliklerine bağlı olarak sahip oldukları ilerleme hızıdır (Zeren, 1995). (Şekil 2.2)



Şekil 2.2 Ses dalgasının şematik gösterimi

Bu özellikler içerisinde bizi en çok ilgilendiren iki tanesi, ses dalgalarının frekans ve genlik özelikleridir. Bir ses dalgasının bir saniye içerisinde gerçekleştirdiği titreşim sayısı, o ses dalgasının frekans değerini verir. Frekans birimi Hz (Hertz)'tir. Frekans ve periyot arasında basit bir ilişki vardır: frekans = 1 / periyot (periyot, tam bir salınım sırasında geçen zamanı belirtir). Yüksek frekanslar, yüksek tonlu tiz sesleri oluştururlar. Düşük frekanslar, alçak tonlu pest sesleri oluştururlar. Sesin hareketi sırasında ortam içindeki tüm parçacıklar aynı frekansla titreşir. Her parçacık bir diğer parçanın hareketi

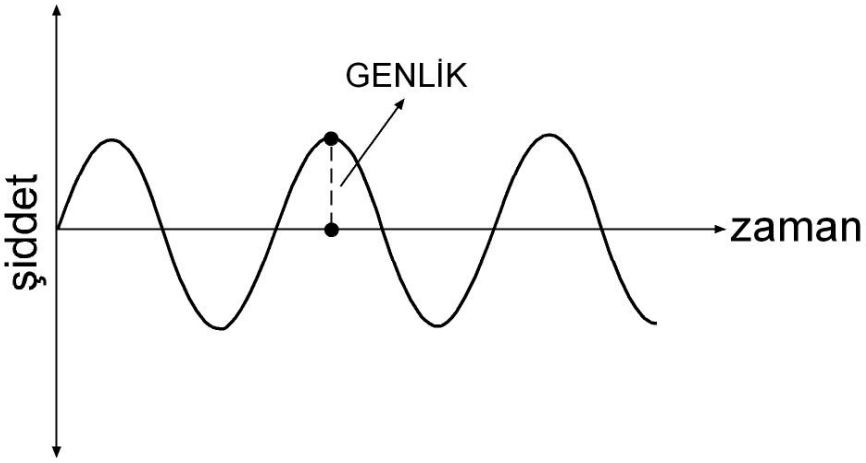
sonucu titreşir. Kulağımıza gelen sesin frekans değeri, aynı zamanda kaynağın da titreşim frekansıdır [11]. (Şekil 2.3)



Şekil 2.3 Frekansı 3 olan bir ses dalgası

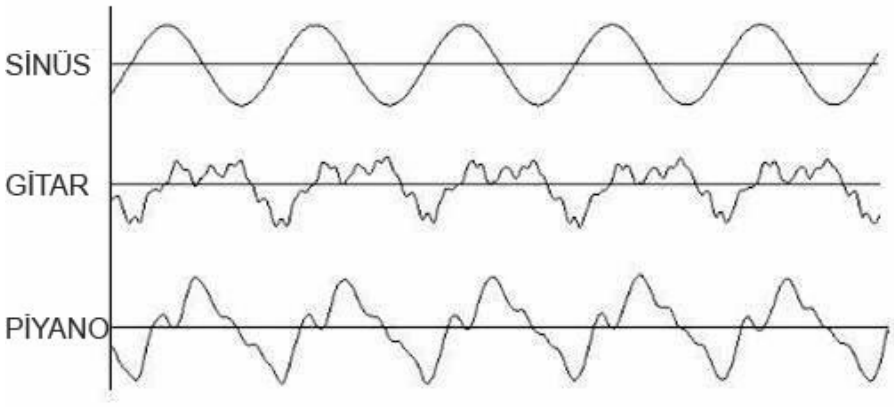
İnsanın 20 Hz ile 20000 Hz arasındaki frekansları duyabildiği kabul edilir. Eğer bir frekans 20 Hz'in altında ise bu tür titreşimlere *"ses altı"* titreşimler, frekans 20 kHz' in üzerinde ise bunlara da *"ses üstü"* titreşimler denilmektedir.

Ses dalgalarının bir diğer özelliği genlik (amplitude)'tir. Basınç genliği, ses dalgalarının atmosfer basıncında oluşturduğu en büyük + ve en küçük - sapma değerleridir. Ses dalgalarını oluşturan sıkışma ve genleşmeler arasındaki fark, dalgaların genliğini belirler. Eğer ses dalgasına fazla enerji yüklenirse, dalga daha büyük bir genlikle titreşir. Ses dalgasındaki titreşim genliği ne kadar fazla ise, ortam tanecikleri (örneğin hava molekülleri) tarafından taşınan enerji de o kadar fazladır. Enerji ne kadar fazla ise, sesin şiddeti (gürlüğü) de o kadar büyük olacaktır. (Şekil 2.4)



Şekil 2.4 Ses dalgalarının genliği

Ses dalgalarının fiziksel özelliklerinden biri de o sesin tınısıdır. Tını, sesin rengini ifade eden bir terimdir. Her sesin kendine özgü bir tınısı vardır. Ses tınıları, insanın beyin ve kulak ikilisinin algı ve yorumu ile ayırt edilebilmesine rağmen, tınıların sayısal bir ifade olarak karşılığı yoktur. Bu durum, sesleri tınısından yola çıkarak tanıyabilecek bir aletin bugüne kadar icat edilememesine etkendir (Sözen, 2003). (Şekil 2.5)



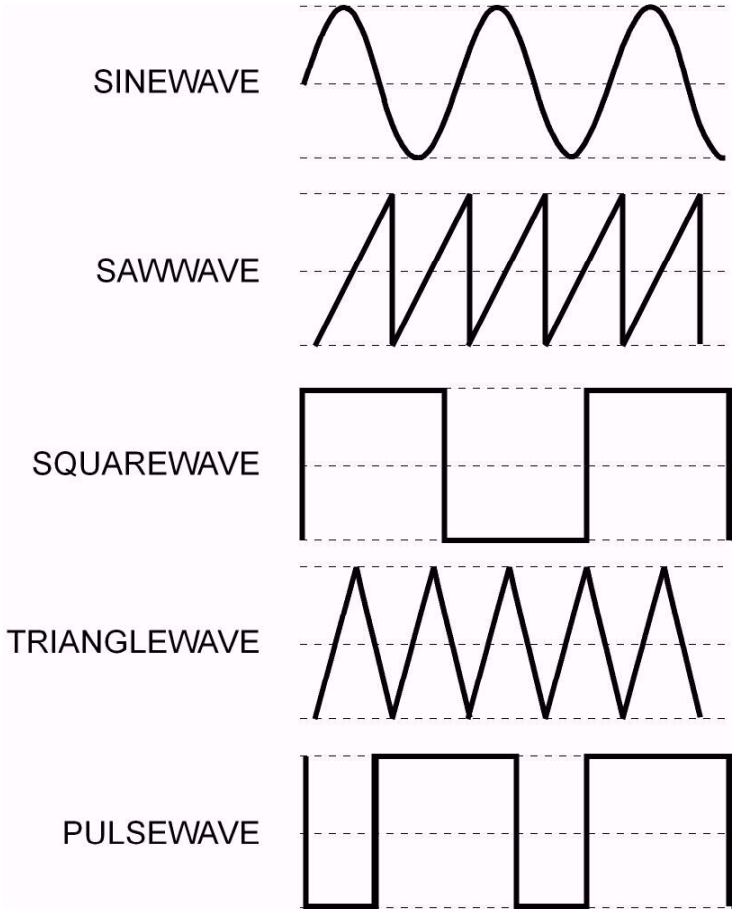
(Kaynak: www.bgst.org/muzik/egitim/muzikfizigi.html)
Şekil 2.5 Seslerin tınılarına göre farklılık gösteren ses dalgaları

Şekil 2.5'te görünen ses dalgalarından olan sinüs dalgası tekdüze bir sestir. Titreşimleri basit bir sinüs eğrisi karakteri taşıyan seslere *"saf ses"* denir. Ancak bu sesler doğada bulunmaz. Doğada duyduğumuz sesler, aslında basit seslerden oluşmuş olan bileşik seslerden başka bir şey değildir. Şekil 2.5'te gösterilen gitar ve piyanoya ait ses dalgası formları bu bileşik seslere örnek teşkil ederler. Bu şekilde görüldüğü gibi karmaşık bir

yapıya sahip periyodik ses dalgalarını oluşturan, farklı frekanslardaki sinüs dalgalarına *"harmonik"* adı verilir. Bir periyodik dalga sonsuz tane farklı frekansta dalgaların toplamından oluşur.

Aynı zamanda ses dalgaları sahip oldukları eğrinin şekline göre de ayrılırlar. Bu noktada tez konumuz için en önemli olan ses dalgası türleri, sinüs dalgası, testere dalgası, kare dalga, üçgen dalga ve darbeli dalgadır [12]:

- Sinüs dalgası (sinewave), dalga formunun sahip olduğu şiddet değerinin zamana bağlı olarak trigonometrik sinüs fonksiyonunu izleyerek devam ettiği dalga türüdür.
- Testere dalgası (sawwave), dalga formu testere şekline benzeyen, şiddet değerinin zamana bağlı olarak sürekli yükseldiği ve ani olarak en düşük değere geri döndüğü bir dalga türüdür.
- Kare dalga (squarewave), en genel olarak elektronik ve dijital sinyal işleme alanında kullanılan sinyal türüdür. Birbirini düzenli bir şekilde izleyen ve ani olarak değişen iki değerin arka arkaya gelmesi ile oluşur.
- Üçgen dalga (trianglewave), üçgen biçimli dalga formuna sahiptir. Sürekli olarak yükselen ve alçalan bir şiddet değerine sahiptir. Kare dalga kadar sert olmayan ve sinüs dalgası kadar da yumuşak olmayan bir ses üretir.
- Darbeli dalga (pulsewave), dalga formu olarak kare dalgaya benzer. Ancak kare dalga gibi bir simetrik alçalış ve yükseliş süresine sahip değildir. (Şekil 2.6)

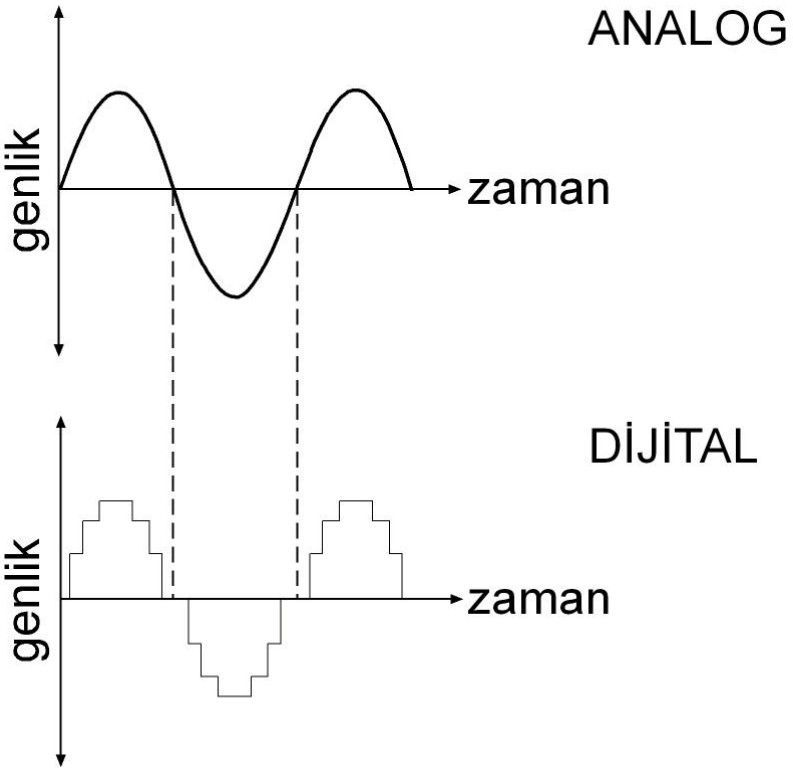


Şekil 2.6 Farklı dalga formları

2.1.2 Analog ve Dijital Ses Teknolojisi

Sesin saklanması ve iletimi için iki farklı teknik kullanılır. Bizim çalışmamız açısından bu iki farklı tekniğin incelenmesi gerekmektedir. Bunlar analog ve dijital tekniklerdir.

Günümüzde bu tekniklerin her ikisinin de farklı ortamlarda kullanıldığına tanık olmaktayız. Analog terimi, verilerin sürekli bir fiziksel değişkenle gösterimini; dijital terimi ise, veri kaydının sayısal olarak gerçekleştirilmesini işaret etmektedir (Sankur, 2004). Analog veri ile dijital veri arasındaki en büyük fark, analog verinin sürekli olan bir ölçekte, dijital verinin ise rakamlarla sınırlı olan ve sürekli olmayan bir ölçekte var olmasıdır. (Şekil 2.7)



Şekil 2.7 Analog ve dijital veriler

Analog sinyaller bilgiyi sürekli değişen bir değerle temsil eder ve sonsuz sayıda durumu temsil edebilir. Analog veriler, giriş sinyallerinin başka sinyaller ile elektriksel işlemlerden geçirilmesi ile elde edilir. Bu durum analog sinyallerde, giriş ve çıkış değerleri arasında matematiksel bir formül ile gösterilebilir bir ilişki olduğunu gösterir (Hawksford, 1991). Analog sinyal, kaynağına en yakın sinyal türüdür.

Analog ses teknolojisi ile günlük hayatımızda en çok karşılaştığımız alanlardan biri teyp teknolojisidir. Teypler, bir sesi alıp aslına en yakın şekilde manyetik bandlara kaydeden ve bu sesi yine okuyup dışarı verebilen cihazlardır. Bir teybin işlevini yerine getirmesini sağlayan en temel bileşeni teyp kafasıdır. Teyp kafasında kaydedici, okuyucu ve silici bulunur. Teyp kafaları manyetik bandlara etki ederek çalışırlar. Teyp bandları ince ve özel bir tabana sahip olması gereken manyetik şeritlerdir (Sözen, 2003). (Şekil 2.8)

(Kaynak www.wikipedia.org)

Şekil 2.8 Kaset biçiminde sarılmış teyp bandı

Birden fazla kez kayıt yapılabilen bandlar, kapalı kaset biçiminde sarılan şekilleri ile çok kullanılmaktadır. Ancak dijital teknolojilerin zamanla gelişmesi teyplerin ve teyp bandlarının kullanımını azaltmıştır.

Seslerin kayıt ve saklanması işlemlerinde kalitenin yükseltilmesi dijital teknolojiye geçişle hızlanmıştır. Analog sistemlerde ses sinyallerinin dalgalar şeklinde alınıp kaydedilmesine karşın, dijital sistemlerde veriler sayısallaştırılarak kaydedilip okunmaktadır. (Şekil 2.7) Analogdan dijitale çevirme işleminde ses kalitesinde kayıp gerçekleşir ancak dijital seslerin bir ortamdan diğerine aktarılmasında kayıp olmaz. Dijital sesler, bilgisayarda herhangi bir veri gibi ele alınabilmekte ve işlenebilmektedir. Dijital veri, rakamların arka arkaya dizilmesi ile elde edildiği için üzerinde biçim değişikliği işlemi yapılabilir. Bozulmuş bir dijital veriye hata düzeltme kodları eklenerek düzeltilmesi sağlanabilir. Ayrıca, dijital verilerin saklanma koşulları analog verilerden çok daha kolaydır. Tüm bu avantajlar doğrultusunda, dijital veri birçok alanda olduğu gibi müzik ve ses teknolojileri alanında da analog verinin yerini almaktadır. Bizim çalışmamız açısından da, dijital verilerin bilgisayar teknolojisi ile bu bütünleşik çalışabilme özelliği önemli bir avantajdır.

Dijital teknolojilerin genişlemesi, eski analog manyetik bandlı teyp kasetlerinin yerini dijital ses bandlarının, cd (compact disc)'lerin almasını sağlamıştır. Eski analog sistemlere göre, daha geniş bir veri saklama ve hata düzeltme imkanına sahip bu dijital

teknolojiler, bilgisayar teknolojisi ile bütünleşik çalışma niteliği sayesinde de tercih edilir durumdadır. (Şekil 2.9)

(Kaynak: www.wikipedia.org)
Şekil 2.9 Cd (compact disc)

Bilgisayarların ucuzlaması ve ses üretimine yönelik donanım ve yazılımların çoğalması dijital ses kullanımını sınırsız bir boyutta genişletmiştir. Ses ve müzik parçalarına her türlü düzenlemeyi yapabilmek ve doğal olmayan sentetik sesleri üretmek gibi avantajları ile bilgisayarlar, birçok ses uygulaması için vazgeçilmez konuma gelmiştir. Bilgisayar teknolojisiyle bütünleşik çalışma imkanına sahip olan ve dijital teknolojinin gelişmesi ile ses ve müzik alanında kullanılmaya başlayan diğer teknolojiler arasında, synthesizer (ses birleştirici), equalizer (eşitleyici), mixer (karıştırıcı), sampler (örnekleyici) sayılabilir.

Tasarladığımız araçta ses üretimi bilgisayar ile gerçekleştirildiği için dijital seslerin burada anlatılan özellikleri çalışmamızda önem teşkil etmektedir.

2.1.3 Hoparlörler

Bir hoparlör, elektrik sinyallerini sese dönüştüren elektromekanik bir dönüştürücüdür. Hoparlörler çalışma şekli olarak mikrofonlara benzerler ancak hoparlörlerde işleyiş ters tarafa doğrudur. Mikrofonlar ses titreşimlerini elektrik sinyallerine dönüştürürken, hoparlörler tam tersi olarak, elektrik sinyallerini ses titreşimlerine dönüştürürler. Bir hoparlör yapısal olarak; metal kasnak, mıknatıs, bobin ve metal kasnağa tutunan bir

zardan oluşmaktadır (Sözen, 2003). Ses frekansına sahip sinyal hoparlörün bobinine uygulanır. Böylece bobin içerisinde ses frekansında bir manyetik alan oluşur. Bu değişken manyetik alan, bobin içinde bulunan mıknatısa aynı frekansla değişen, itip çekme şeklinde bir kuvvet uygulayarak mıknatısı ve ona bağlı diyaframı titreştirir, böylece ses dalgaları üretir. Cihazın koni biçimindeki parçası da bu ses dalgalarını belli bir yöne göndermeye yarar (Gayford, 1970). (Şekil 2.11)

(Kaynak: www.wikipedia.org)
Şekil 2.10 Hoparlör

Bir hoparlörün niteliğini belirleyen üç değer vardır:

- Empedans: Bobinin uygulanan sinyalin değişkenliğine karşı gösterdiği dirençtir. Bu değer bobinin sargı sayısı ile büyür.
- Güç: Bobinden birim zamanda yayılan en büyük ses enerjisidir.
- Çap: Bir hoparlörün çapı küçüldükçe tiz sesleri, çapı büyüdükçe pest sesleri daha başarılı verir (bas, medium, tweeter vb.)

Kamera kaynaklı işitsel sanat aracının ses ile ilgili teknik işlemlerinin son aşamasını, bilgisayar tarafından gönderilen elektrik sinyallerinin hoparlörler aracıyla ses dalgalarına dönüştürülmesi işlemi oluşturur. Bu yüzden araçtan alınacak performans, sisteme bağlanmış olan hoparlörlerin ses frekanslarını oluşturma duyarlılığına paralel olarak artacaktır. Aracın oluşturduğu seslerin kullanıcının amaçlarına paralel bir şekilde oluşturulması, seçilen hoparlör ile de ilgili olacaktır.

2.1.4 Renklerin Fiziksel Özellikleri

Renk, insan gözünün görebildiği ışık tayfının belirli bir elektromanyetik dalga boyudur. Çevremizdeki nesnelere yeni bir boyut kazandıran renk gerçekte ışıktır. Nesnenin üzerine düşen ışığın bir bölümü (belli dalga boyundakiler) nesne tarafından emilir, yansıyan ışık olarak gözümüze gelen ışıklar ise renktir (Kılıç, 1994). (Tablo 2.1)

Tablo 2.1 Renklerin dalga boyları ve frekansları

Renk	Dalga boyu	Frekans
Kırmızı	~ 625-740 nm	~ 480-405 THz
Turuncu	~ 590-625 nm	~ 510-480 THz
Sarı	~ 565-590 nm	~ 530-510 THz
Yeşil	~ 500-565 nm	~ 600-530 THz
Siyan	~ 485-500 nm	~ 620-600 THz
Mavi	~ 440-485 nm	~ 680-620 THz
Mor	~ 380-440 nm	~ 790-680 THz

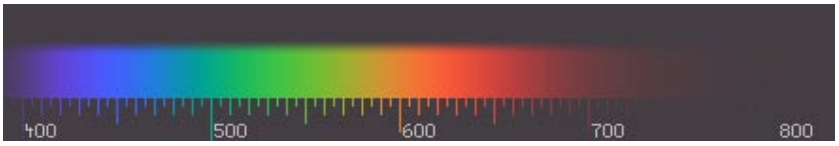
(Kaynak: www.wikipedia.org)

İnsan gözü 380nm ile 780nm arasındaki dalga boylarını algılayabilmektedir. Bu sebepten dolayı elektromanyetik spektrumun bu bölümüne *"görünen ışık"* denir.

Renklerin üç temel niteliği vardır. Bunlar:

- Temel renk (hue)
- Doygunluk (saturation)
- Parlaklık (brightness)'tır.

Temel renk, renk türünü belirler. Mavi, kırmızı, yeşil gibi renk yelpazesindeki değişik dalga boylarıdır. (Şekil 2.11)



(Kaynak: www.wikipedia.org)
Şekil 2.11 Renklerin türünü belirleyen dalga boylarının oluşturduğu skala

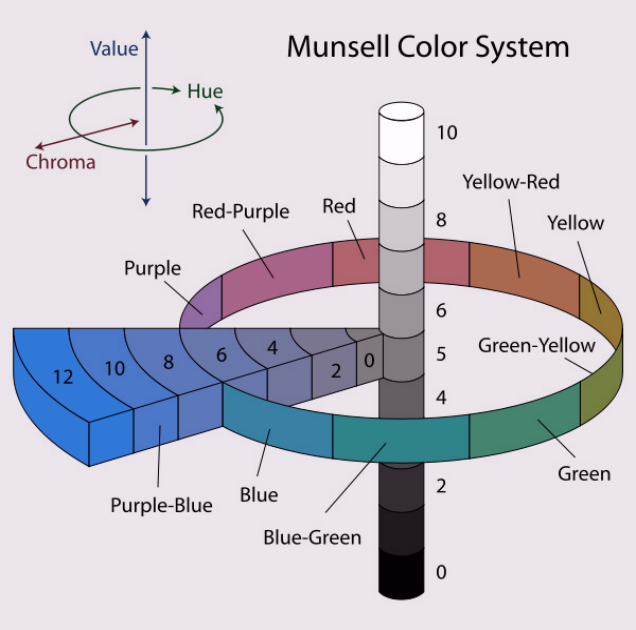
Doygunluk, renklerin saflık derecesini belirler, en saf renk, rengin kendisidir. Doygunluğu yüksek olan bir renk, doygunluğu azaldıkça griye doğru yaklaşacaktır. (Şekil 2.12)


Şekil 2.12 Kırmızı renginin doygunluk skalası

Parlaklık rengin koyu ve açık olması ile ilgilidir. En açık renk beyaz, an koyu renk siyahtır. Bir renk, parlaklığını arttırdığımızda beyaza doğru, parlaklığını azalttığımızda ise siyaha doğru yaklaşır. (Şekil 2.13)


Şekil 2.13 Mavi rengin parlaklık skalası

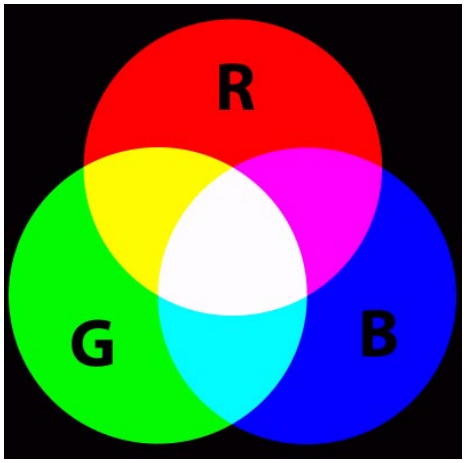
Renklerin belirlenmesini sağlayan bu üç özelliğin (temel renk, doygunluk, parlaklık) bir arada gösterildiği ve üç boyutlu olarak temsil edildiği bir renk skalası sistemi, yirminci yüzyılın başlarında Albert H. Munsell tarafından başarıyla oluşturulmuştur [13]. (Şekil 2.14)



(Kaynak: www.wikipedia.org)

Şekil 2.14 Munsell'in oluşturduğu renk sistemi

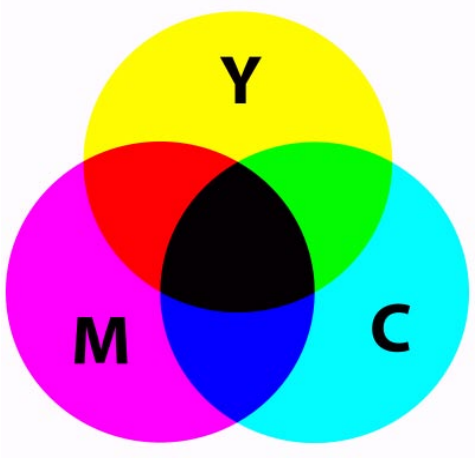
Doğada görünen bütün renkler birçok farklı rengin karışımından oluşmaktadır. Renklerin fiziksel karışımı *"toplamalı"* ve *"çıkarmalı"* olmak üzere iki şekilde gerçekleşir (Kılıç, 1994). Toplamalı karışım ile anlatılmak istenen değişik renk frekanslarının birleşerek gözümüze ulaşmasıdır. Bu karışımda üç ana renk (kırmızı, yeşil, mavi) birleşerek farklı renkleri oluştururlar. Kırmızı ile yeşilin birleşmesinden sarı, kırmızı ile mavinin birleşmesinden eflatun, mavi ile yeşilin birleşmesinden siyan rengi oluşur. Bu üç renk tam olarak birleştiğinde ise beyaz elde edilir. Bu renkler kendi aralarında bu şekilde birleşerek beyaz rengi elde ettikleri için toplamalı renk karışımı denmektedir. Bu karışım şekli ışık ortamı ile ilgili olan bir karışımdır. Bu yüzden bilgisayar görüntüleri için bu karışım şekli geçerlidir. (Şekil 2.15)



(Kaynak: www.wikipedia.org)

Şekil 2.15 Işık ortamında geçerli olan toplamalı renk karışım şekli

Renklerin diğer karışım şekli çıkarmalı olan sistemdir. Bu sistem kimyasal ortamda geçerli olan bir karışım şeklidir. Bu karışım şeklinde kullanılan üç ana renk (sarı, eflatun, siyan) birleştiklerinde siyahı oluştururlar. Siyahtan eflatun (magenta) çıkarılırsa yeşil, siyahtan sarı çıkarılırsa mavi, siyahtan siyan (cyan) çıkarılırsa kırmızı renk elde edilir. Diğer bütün renkler siyahtan bazı renklerin çıkartılması ile elde edilirler. Bu yüzden bu karışım şekli çıkarmalı ismini almıştır. (Şekil 2.16)



(Kaynak: www.wikipedia.org)

Şekil 2.16 Kimyasal ortamında geçerli olan çıkarmalı renk karışım şekli

Renklerin sahip olduğu bu fiziksel özellikler, geliştirdiğimiz araç için önem arz eden noktalardır. Görüntülerin analiz edilmesi dediğimiz şey, genel anlamıyla renklerin ayrıştırılmasıdır. Tez çalışmamız, ışığın kamera ile algılanıp bilgisayar ortamında

işlenmesini de konu ettiğinden, renklerin kimyasal karışımlar ile elde edilme yönteminden çok ışıklar ile elde edilme yöntemi bizim için daha önemlidir.

2.1.5 Kameralar

Tasarladığımız aracın temel özelliği olan görüntülerin işlenmesi ile seslerin oluşturulması işlevi, görüntülerin kamerayla yakalanmasıyla başlar. Bu durum, kamerayı işitsel-sanat aracı mekanizmasının önemli bir bileşeni durumuna getirir.

Sistemin çalışması için kullanılabilecek olan kameralar temelde iki gruba ayrılır. Bunlar, analog ve dijital kameralardır. Analog video kameralar standart olarak analog görüntü sinyalleri oluştururlar. Bu sinyalleri bilgisayarda işlenebilir dijital sinyallere dönüştürebilmek için bir takım ara dönüştürücüler kullanmak gerekir (PCI ya da PCI Express kartları, harici USB ya da Firewire arayüzleri vb.). Görüntü kalitesi açısından analog kameraların yakaladığı görüntüler genelde tatmin edici olur. Diğer taraftan kablo uzunlukları daha az kısıtlayıcıdır. Diğer taraftan sahne performansı ve benzeri gerçek zamanlı uygulamalarda veri aktarmada gecikme süresi daha az yaşanabilir [14]. (Şekil 2.17)

(Kaynak: www.wikipedia.org)
Şekil 2.17 Analog kamera

İşitsel-sanat aracının mekanizmasında, analog kameralar dışında kullanılabilecek olan diğer kamera çeşidi dijital kameralardır. (Şekil 2.18)

(Kaynak: www.wikipedia.org)
Şekil 2.18 DV video kamera

Dijital kameralar farklı kaliteler ve farklı maliyetler yelpazesine sahiptir. Bunlar içerisinde görüntü kalitesi ve maliyeti en yüksek olanları DV video kameralarıdır. Dijital kameralar genellikle firewire çıkışları olan ve masaüstü ya da dizüstü bilgisayara direk olarak bağlanabilen kameralardır. Görüntü kaliteleri yüksektir. Ancak genelde yüksek çözünürlükte çalıştıklarından dolayı aktarımda gecikme sorunu yaşanabilir. Aynı zamanda ucuz olan bazılarında ise, kamera otomatik odaklamayı kapamaya izin vermiyor olabilir. Bu da istenmeyen görüntü hareketleri doğuracağından dolayı sorun teşkil edecektir. Aynı zamanda kablo uzunlukları bu kameraların kısıtları arasındadır.

Dijital kameralar ile ilgili bir diğer seçenek webcam'lerdir. Bu kameralar diğer seçeneklere göre daha ucuz bir çözümdür. Aynı zamanda herhangi bir donanımsal destek gerektirmezler. Bilgisayara herhangi bir USB portundan bağlanabilirler. Ancak diğer kameralara göre daha kötü bir görüntü kalitesine sahiptirler. Bu durum kullanıcının kendi kontrollerini nasıl bir ortamda değerlendireceğine bağlı olarak sorunlar yaratabilir. Aynı zamanda yine otomatik odaklama özelliği birçoğunda kapatılamaz ve bu da istenmeyen görüntü hareketlerine yol açabilir [14]. (Şekil 2.19)

(Kaynak: www.wikipedia.org)
Şekil 2.19 Webcam

Görüldüğü gibi tasarladığımız aracın önemli bir bileşeni olan kameralar farklı özellikler sunan seçeneklere sahiptir. Bütçeye, istenen görüntü kalitesine, donanımsal bağlantı imkanlarına, gecikme süresinin önemine ve kullanılacak olan mekanın özelliklerine göre bu seçeneklerden biri ön plana çıkabilir. (Tablo 2.2)

Tablo 2.2 Kameraların karşılaştırılması

Kameralar	Avantajlar	Dezavantajlar	Maliyet
Analog Kamera	- Kaliteli görüntü - Hızlı aktarım - Çeşitli lens imkanları - Farklı kablo uzunlukları	- Verileri dijital formata dönüştürmek için ara elemanlar gerekli	- 100 – 800 $
DV Video Kamera	- Yüksek görüntü kalitesi - Bilgisayara direk bağlantı - Çeşitli lens imkanları	- Yüksek maliyet - Yüksek veri yoğunluğunda gecikme - Kablo uzunluğu kısıtlı	- 500 – 5000 $
Webcam	- Düşük maliyet - Bilgisayara direk bağlantı	- Düşük görüntü kalitesi - Tek lens - Kablo uzunluğu kısıtlı - Bazılarında odaklama sorunu	- 20 – 100 $

(Kaynak: eyecon.palindrome.de)

Çalışmamıza konu olan işitsel-sanat aracını geliştirirken kullanacağımız kamera webcam olacaktır. Bunun nedeni düşük maliyeti ve bilgisayar ile bağlantı konusunda kolay bir kuruluma sahip olmasıdır. Ancak tasarlanan aracın performansının geliştirilebilmesi için, bu seçimin diğer kamera seçeneklerinin de gözetilerek yapılması gerekmektedir.

2.2 YÖNTEM

Bu kısımda, daha önce anlatılan malzeme ve materyalleri hangi yöntem ve teknolojiler ile işleyeceğimizi inceleyeceğiz. Öncelikle araç geliştirme ortamının özelliklerini ve neden seçtiğimizi, ardından işitsel-sanat aracının, hangi görüntü ve ses özellikleri üzerine oturacağını belirleyeceğiz.

2.2.1 Araç Geliştirme Ortamı

2.2.1.1 Temel nitelikler

Tarafımızdan geliştirilen yazılım, ses ve görüntü işlemlerine ve bir grafik arayüz ile kullanıcı kontrollerinin sağlanmasına dayanan bir sistemdir. Oluşturduğumuz yazılımın en temelde şu işlevleri yerine getirilmesi beklenmektedir:

- Kamera aygıtı ile iletişim kurabilmesi ve gerçek zamanlı görüntü alabilmesi
- Kullanıcı kontrolleri için grafik arayüz oluşturabilmesi
- Kullanıcı kontrolleri ile parametreleri belirlenebilen ve değiştirilebilen ses frekanslarının oluşturulabilmesi

Bu işlevleri yerine getirecek olan yazılımın hangi programlama dili ve geliştirme ortamı ile gerçekleştirileceği önemli bir karar noktasını oluşturur. Yazılımı geliştirecek olan kişi, programlama dilleri hakkında temel bilgilere sahip olarak bunlar arasında hangisinin kullanılacağına kesin ve açık olarak karar vermelidir. Bizim bu kararı

verirken özellikle göz önünde tuttuğumuz nitelikler şunlar olacaktır (Karahoca ve diğ, 1998):

1) Uygunluk: Kullanacağımız programlama dilinin, geliştireceğimiz yazılımın özel ve genel işlevlerinde başarı ile kullanılabilir olması gerekmektedir.
2) Karmaşıklık: Programlama dili, yazılımın gerekli kontrol yapılarını, ihtiyacı olan veri yapılarını ve gerekli işlemleri rahatlıkla yapabilecek bir karmaşıklığa sahip olmalıdır.
3) Organizasyonel Düşünce: Kullanılacak olan programlama dili belirlenirken, yazılımı geliştirecek olan kişi ya da kişilerin hangi dillere hakim olduğu ya da bu kişiler tarafından hangi dillerin çabuk öğrenilebilir olduğu düşünülmelidir.
4) Destek: Seçilecek olan programlama dilinin daha önce benzer çalışmalarda başarıyla kullanılmış olmasına dikkat edilmelidir. Bu aynı zamanda bu dilin daha önceden birçok sefer kontrol edilmiş olmasını ve hatalarının daha çok ayıklanmış olmasını sağlar. Aynı zamanda herhangi bir sorunla karşılaşıldığında sorunu çözebilecek çevrelerin olması önemlidir.
5) Etkinlik: Yazılımın performans ihtiyaçlarını karşılayabilecek bir programlama dili gereklidir. Yazılım standart bir PC üzerinde geliştirilecektir. Dolayısı ile işlemci ve bellek kullanımı ve derleyicinin performansı kullanılan programlama dilinde oldukça iyi yönetilebilir olmalıdır.

Yazılımı geliştirmek için belirlediğimiz programlama dilini seçerken cevaplamaya çalıştığımız temel ölçütlerimiz bunlar olmuştur.

2.2.1.2 Nesneye dayalı programlama

Nesneye dayalı programlama çalışmaları 1970'lere dayanmaktadır. Özellikle görsel arayüzlerin gelişmesi nesneye dayalı programlamayı önemli bir noktaya getirdi. Bu özelliği gösteren programlama dillerinin temel yapıları sınıflardan ve bunlardan türetilen nesnelerden oluşmasıdır. Her bir sınıf kendi içerisinde tanımlanmış olan yöntemler barındırır ve bu sınıflardan türetilen bütün nesnelerin içerisindeki yöntemler aynı şekilde çağrılsalar bile farklı sonuçlar verirler. Nesnelerin içerisindeki verilere ve metotlara ayrı ayrı müdahale edebildiğimizden dolayı, biri üzerindeki etkilerimiz diğerlerini etkilememektedir. Aynı zamanda, program kodu her nesne için tekrar

yazılmamaktadır. Bir sınıfa ait olan bir nesne o sınıf içerisinde belirlenmiş olan bütün veri yapılarını, metotlarını ve özelliklerini kendi içerisinde de taşımaktadır (Karahoca ve diğ, 1998).

Bu özellikler bizim için önemlidir. Çünkü birden fazla kanal görüntü ve sesi oluşturması ve yönetmesi gereken yazılımın, bunları birer nesne olarak tanımlaması ve özelliklerini belirleyebilmesi gerekmektedir. Böylece her bir ses ve görüntü kanalı ayrı olarak, aynı anda işlenebilecektir.

Nesneye dayalı programlama dilini bizim için önemli hale getiren bir özellik de, programlama kodlarındaki işlevselliğe bağlı olarak kod yazım sürecinin daha verimli geçirilmesidir. Her bir görüntü ve ses kanalına gereken nesneler için ayrı ayrı her bir özelliği tanımlamak yerine, hepsinin ortak noktalarını tek bir seferde sınıf olarak tanımlamak bizim için önemli bir zaman ve emek kazancıdır.

Nesneye dayalı programlamayı bizim için kaçınılmaz kılan bir özellik de, geliştirdiğimiz yazılımın görsel bir arayüze sahip olmasıdır. Arayüz ve içerisinde kullanılacak olan elemanlar, çalışmanın teknik doğası gereği birer nesne olarak tanımlanacaktır.

Bütün bu özellikler ve gereksinimler, yazılımımızın nesneye dayalı bir programlama dili ile gerçekleştirilmesini dayatmaktadır.

2.2.1.3 Java

Nesneye dayalı programlama seçeceğimiz dilin temel özelliği olacaktır. Yukarıda belirlediğimiz temel işlevler ve ölçütler de bu programlama dili ile karşılanabilmelidir.

Belirlediğimiz temel işlevleri gerçekleştirebilen bir ses uygulaması yazılımının geliştirilebilmesi için yukarıdaki özelliklere sahip olması gereken programlama dilinin önemli niteliklerinden bir tanesi, kayan nokta hesaplamalarını yapabiliyor olmasıdır (Dannenberg ve diğ., 1996).

Bu özellik bizim için önemlidir. Tamsayı hesaplamaları ile çalışan bir programlama dili daha hızlı çalışabilir. Ancak gerçekleştirdiğimiz yazılımın ses ve görüntü işlemek gibi işlevlere sahip olduğu düşünülürse, kayan nokta hesaplamalarının duyarlılığı kaçınılmaz olacaktır.

Özellikle yukarıda belirlediğimiz beş temel ölçütün (uygunluk, karmaşıklık, organizasyonel düşünce, destek, etkinlik) gösterdikleri, nesneye dayalı bir programlama dili olması, kayan noktalı hesaplamalara ve görsel elemanların yaratılmasına imkan vermesinden dolayı, Java programlama dili, bu yazılımın işlevlerinin yerine getirilebilmesi için temel oluşturacak programlama dili olarak belirlenmiştir.

Java platformu hem bir programlama dili, hem de bir ortam olarak düşünülebilir. Açık kodlu, nesneye yönelik, güvenli ve sağlam bir programlama dilidir. Java platformu, geliştirilen uygulamaların farklı işletim sistemleri üzerinde çalıştırılabilmesi düşüncesi ile geliştirilmiş bir teknolojidir. Java uygulamaları, JVM (Java Virtual Machine) tarafından yorumlanır. JVM, işletim sisteminin üstünde çalışır [15]. Bu özellik, geliştirdiğimiz yazılımın farklı ortamlarda çalıştırılabilir olmasını da sağlayacaktır.

2.2.1.4 Processing

Java programlama dili yukarıda bahsedilen nedenlerden dolayı geliştirdiğimiz yazılımın iskeletini oluşturmuştur. Ancak Java ve diğer geleneksel programlama dilleri (C, C++ vd.) bir çok ses ve görüntüyü birlikte işleme ve etkileşimleri uygulamaya katma konusunda bir takım zorluklar yaratabilmektedir (Dannnenberg, 2002). Özellikle kamera gibi harici aygıtların sisteme bağlanması genelde bazı ek çalışmaları gerektirmektedir. Bu zorlukların aşılabilmesi için Java tabanlı çalışan (dolayısıyla daha önce bahsettiğimiz bütün avantajları kapsayan) bir ortam olan Processing kullanılacaktır.

Processing; imajlar, animasyonlar ve etkileşimler oluşturmak isteyenler için geliştirilmiş bir açık kaynak programlama dili ve ortamıdır. Özellikle sanatçılar, tasarımcılar, öğrenciler, araştırmacılar için geliştirilmiş bir ortamdır. Bilgisayar programlamasının görsel amaçlı ve çeşitli çoklu ortam uygulamalarında profesyonel amaçlı olarak

kullanılması için yaratılmış bir araçtır. Temel yapısını java dili oluşturur. Ancak sınıfların tanımlanması, program kodlarının oluşturulması ve çevre aygıtlar (kamera, mikrofon, hoparlör, sensör) ile bağlantı kurulması kolaylaştırılmış bir araçtır [16].
Çevre aygıtlarla iletişim kurulması, görüntü işleme, seslerin yaratılması, arayüzün oluşturulması ve etkileşimlerin sağlanması gibi amaçlarımız olduğu düşünüldüğünde, processing bizim için oldukça verimli bir çalışma ortamı yaratmaktadır.

2.2.1.5 Video

Yaptığımız tez çalışmasının amacının yani ses kaynaklarının yakalanan görüntüler ile belirlenmesinin gerçekleştirilebilmesi, kameranın sisteme başarı ile tanıtılması ve alınan görüntülerin işlenebilir formata dönüştürülebilmesine bağlıdır. Bunun için kamera verileri ile iletişim kurabilmemizi sağlayan bir sınıfın kullanılması gerekmektedir. Bu noktada, iki seçeneğimiz vardır.

Birinci seçenek, processing için eklenti olarak hazırlanmış olan harici sınıfları kullanmaktır. Bu sınıflar processing ile ilgilenen kişi veya grupların geliştirdiği ve dokümante ettiği sınıflardır. Özellikle görüntü işleme konusunda hazır yöntemler sunmaları ile ilgi çekicidirler. Bu sınıfları kullanmanın bizim için avantajlı yönleri vardır. Hazır olarak geliştirilmiş olmaları ve kolayca çalıştırılabilmeleri bu sınıfların artı yönleridir. Aynı zamanda birçok uygulamada kullanılmış olmaları güvenilirlik açısından da az sorun yaşanmasını sağlamaktadır. Ancak bu harici ve hazır sınıfların kullanımlarının bizim için dezavantajlı yanları da vardır. Özellikle arka planda çalışan kodun nasıl bir işlem sırası izlediği bilinememektedir. Gereksiz olan ancak bizim tarafımızdan algılanamayacak olan işlemler her zaman olasılıklar dahilindedir. Bu durum çalışmanın performansını etkileyebilir. Diğer yandan, görüntü işleme yolları ve çıkarılacak sonuçlar düşünüldüğünde, hazır video sınıflarının özelliklerine uygun olması için, mimarisini oluşturduğumuz sistemde değişikliklere gidilmesi gerekebilir. Bu durumlar göz önüne alındığında kendi özgün amacımız için tatmin edici sonuçlar veren ve sadece hedefimizle ilgilenen bir anlayışla yola çıkmak daha verimli olacaktır.

İkinci seçenek, processing ortamının içerisinde bulunan dahili sınıfları kullanmaktır. Bu durumda, her ne kadar kamera verisi ile ilişki kurmak ve görüntü işlemek için hazır

kodları kullanamayacak olsak da, yazdığımız kodların işleyişine hakim olacak olmamız büyük bir avantajdır. Sadece kendi özel hedefimize yönelik olan ve bu hedef dışında başka hiçbir şeyle ilgilenmeyen görüntü işleme yöntemleri geliştirmek bizim için daha verimli sonuçlar yaratacaktır. Böylece, kodların hangi işlemleri gerçekleştirdiği ve bunlardan çıkarılan sonuçların güvenilirliği tarafımızca bilinebilecektir. Bu nedenle, kameradan görüntü alıp bunları görüntü işlemlerinden geçirmek için processing ortamının dahili video kütüphanesinin sınıflarının kullanılmasına karar verilmiştir.

Processing ortamında bulunan video kütüphanesi; QuickTime video dosyalarını göstermek, video verilerini almak ve çalışan kodlarla QuickTime dosyaları oluşturmaya izin verir. Bunların yanında bu sınıf, USB ve firewire çıkışlı kameralar ile bağlantı kurulmasını ve görüntülerin yakalanmasını sağlar. Bu kütüphanenin çalışması için QuickTime programının bilgisayara yüklü olması gerekmektedir [17].

Görüntülerin yakalanması ve bunların işlenmesi için gerekli olan kod ve algoritmaların tarafımızdan oluşturulması sırasında kullanılacağımız dahili video kütüphanesinin içerisinde iki adet sınıf bulunur. Bunlar Movie ve Capture sınıflarıdır.

Movie sınıfı, özellikle QuicktTime dosyalarının yüklenmesi, oynatılması, hızının ayarlanması gibi işlemler için kullanılır. Bu amaçla sınıf içerisinde tanımlanmış read( ), available( ), play( ), pause( ), stop( ), loop( ), noLoop( ), jump( ), duration( ), time( ), speed( ), frameRate( ) gibi fonksiyonlar vardır. Tez çalışmamız içerisinde QuickTime dosyaları ile çalışmadığımız için movie sınıfından üretilmiş nesnelere ihtiyaç duymamaktayız.

Video kütüphanesi içerisinde tanımlanmış olan bir diğer sınıf olan Capture sınıfı ise, videolardan ya da sisteme bağlı kamera gibi aygıtlardan görüntü yakalanması için kullanılır. Bu amaçla sınıf içerisinde tanımlanmış list( ), format( ), source( ), settings( ), read( ), available( ), frameRate( ), crop( ), noCrop( ), stop( ) gibi fonksiyonlar tanımlanmıştır. Bizim amacımız gereği, kameradan alınan görüntünün işlenebilmesi için Capture sınıfını ve bu sınıftan türetilmiş olan bir adet nesneyi kullanmamız gerekmektedir.

2.2.1.6 Minim

Minim, processing ortamında kullanılmak üzere JavaSound API'leri ile geliştirilmiş harici bir ses kütüphanesidir. Rahat anlaşılabilir ve kolay uygulanabilir olmasına çalışılmıştır. Özellikle ses çalışmalarının diğer uygulama ortamlarına rahatça adapte edilebilmesi için geliştirilmiştir. Minim, içerisinde dört ayrı paket barındırır. Her bir paket farklı amaçlar için bir araya getirilmiş olan sınıflardan meydana gelir. Bunlar, ddf.minim, ddf.minim.analysis, ddf.minim.effects ve ddf.minim.signals paketleridir [18].

Bu paketlerden ilki olan ddf.minim paketi, herhangi bir Minim uygulaması için gerekli olan sınıfları barındırır. Ses örnekleri oluşturma, giriş ve çıkış ses kanalları yaratma ve Minim kütüphanesinin nesnelerinin bağlı olduğu uygulamaları çalıştırma gibi temel sınıfları ve arayüzleri barındırır. Bu paket içerisinde bulunan çok sayıda sınıf ve arayüz içerisinde bizim için önemli olan iki sınıf vardır. Bunlar Minim ve AudioOutput sınıflarıdır. Minim sınıfı, JavaSound API'lerinden istenenlerin yerine getirilmesi gerekli olan bir sınıftır. AudioSample, AudioSnippet ve AudioPlayer gibi sınıflardan türetilmiş nesnelerin kontrollerini gerçekleştirecek fonksiyonlara sahiptir. AudioInput gibi dışarıdan alınan sesler ve AudioOutput gibi dışarı verilen sesler için oluşturulan kanalları yaratacak ve yönetecek fonksiyonlara sahiptir. Minim statik fonksiyonlar barındıran bir sınıftır. Dolayısı ile Minim fonksiyonlarından faydalanabilmek için nesne yaratmak gerekmez. Bizim için en temelde kullanılması gereken iki fonksiyon, start( ) ve stop( ) fonksiyonlarıdır. Bunlar, Minim kütüphanesine ait işlemlerin başlaması ve sonlandırılması için gereklidir. Bizim için önemli bir diğer sınıf metotu ise getLineOut(int type, int bufferSize, float sampleRate, int bitDepth) isimli çıkış kanalı yaratmaya yarayan fonksiyondur.

Bu fonksiyonun ilk parametresi çıkış kanalının türünü belirler. Minim.MONO ve Minim.STERO olmak üzere iki seçenek vardır. Bizim çalışmamız gereği yaratacağımız bütün ses çıkış kanalları Minim.STERO sabitiyle yaratılacaktır. Diğer giriş parametresi bufferSize'dir. Bu değer, her bir ses örneğinin arabellekte (buffer) kaç byte'lık yer kaplayacağını belirler. Bizim çalışmamız için her bir ses kanalının bufferSize değeri 512 olarak belirlenmiştir. Diğer parametre sampleRate değeridir. Bu değer, üretilen ses

örneklerinin kaç parça ile tanımlanacaklarını belirler. Ürettiğimiz bütün ses kanalları için 44100 değeri olmasına karar verildi. Dördüncü parametre olan bitDepth ise üretilen sesin bit derinliğini belirler. Tez çalışmamızda bütün bit derinlikleri 16 olarak belirlenmiştir.

Kullandığımız Minim paketi içerisinde bizi ilgilendiren bir diğer sınıf AudioOutput sınıfıdır. Bu sınıfın nesneleri ses sinyallerinin üretilmesini sağlar. Üretilen her bir getLineOut nesnesi, AudioOutput sınıfından türetilen nesnelerce tutulur ve bu kanallara yüklenen sinyaller bu nesnelerce üretilir. Bu sınıfa bağlı, ses sinyallerini ses çıkış kanalına bağlayan addSignal( ) ve ses kanallarından sinyalleri silen clearSignals( ) gibi bizim çalışmamızda önemli olan fonksiyonlar vardır.

Minim kütüphanesinin bir diğer paketi ddf.minim.analysis'tir. Bu paket, temel olarak dışarıdan alınan seslerin analiz edilmesi ve sinyal özelliklerinin belirlenmesi üzerine kurulu olan sınıfları barındırır. Ancak tez çalışmamızın amacı gereği biz bu paketi kullanmamaktayız.

Minim kütüphanesinde bulunan bir diğer paket ddf.minim.effects'tir. Bu paket, temel olarak ses sinyallerine çeşitli efektler uygulamak için kullanılan sınıfları barındırır. Tez çalışmamız gereği biz bu paketi de kullanmamaktayız.

Minim kütüphanesinin sahip olduğu son paket ddf.minim.signals paketidir. Bu paket, sinyallerin üretilmesini sağlayan sınıfları barındırır ve bu sınıflar çalışmamız için oldukça önemlidir. Bunlar PinkNoise, PulseWave, SawWave, SineWave, SquareWave, TriangleWave, WhiteNoise, ve Oscillator sınıflarıdır. Bu sınıflar farklı türde sinyal dalgaları üretmeye yararlar. PinkNoise sınıfı, pembe gürültü sinyalleri; PulseWave sınıfı, darbeli dalga sinyalleri; SawWave sınıfı testere dalga sinyalleri; SquareWave sınıfı, kare dalga sinyalleri; TriangleWave sınıfı üçgen dalga sinyalleri; WhiteNoise sınıfı, beyaz gürültü sinyalleri üretmeye yarar. Oscillator sınıfı ise diğerlerini kapsayan bir sınıftır. Herhangi bir Oscillator sınıfı değişkeni bütün diğer sinyal sınıfı nesnelerini tutabilir. Bizim çalışmamız açısından bu sinyal türleri önemlidir. Çünkü kullanıcı ses sinyalinin dalga formunu belirleme özgürlüğüne sahip olacaktır.

Oscillator sınıfından üretilmiş herhangi bir nesneye eklenmiş olan ses sinyallerinin değerleri değiştirilebilir. Bunlar frekans değeri, genlik değeri, pan değeri ve portamento değeridir. Bunlar için sırasıyla; bu sınıfın setFreq( ), setAmp( ), setPan( ) ve portamento( ) fonksiyonları kullanılır. Portamento değeri, sinyalin sahip olduğu frekans değerinde bir değişme olduğunda yeni değere kaç milisaniyede ulaşılacağını belirler.

Dikkat edilmesi gereken bir nokta, ddf.minim.signals paketinin sahip olduğu sınıfların yapımcı fonksiyonlarında üç adet parametre kullanılır. Bunlar frekans değeri, amplitude değeri ve sinyalin sampleRate değeridir. Çalışmamızda ürettiğimiz bütün sinyallerin sampleRate değerlerini, ses çıkış kanalının sampleRate değerine (44100) eşitleyerek çalışıyoruz. Sadece PinkNoise ve WhiteNoise sınıflarının yapımcı fonksiyonlarında bir adet parametre bulunur. Bu parametre amplitude değerini belirleyen parametredir.

Minim ses kütüphanesinin önemli bir özelliği, otuz iki adet ses kanalını aynı anda çalıştırabiliyor olmasıdır. Ses çıkış kanalı sayısının otuz iki olması bize sesler ile ilgili olarak kurabileceğimiz etkileşimler imkanını genişletme şansı tanımaktadır. Özellikle, ses ile ilgili uygulamalarda, ses nesnelerinin farklı kombinasyonlar yaratması çalışmaya ilgi çekici bir özellik kazandırabilmek için önemli bir gerekliliktir (Dannenberg ve diğ., 1996).

2.2.2 Ses ve Görüntünün Eşlenmesi

Tasarlanan işitsel-sanat aracında en temel özellik olan görüntü ve sesin eşlenmesi, öncelikle bu ortamların hangi özelliklerinin birbirlerine karşılık geleceğinin belirlenmesini gerektirir. İlk olarak, dijital ortamdaki ses özelliklerinin hangilerinin kullanıcı kontrolüne bağlı olacağı ve hangilerinin kullanıcı kontrolünden bağımsız olacağına karar verilmelidir.

Daha önce incelediğimiz ses sinyallerinin özellikleri; type, bufferSize, sampleRate, bitDepth, dalga formu, frekans, amplitude, pan ve portamento değerleridir. Bu özelliklerden, etkileşim tasarımının net olabilmesi ve sistemin ürettiği seslerin

performansının daha yüksek olması için kullanıcı kontrollerine imkan tanımadan belirlenecek olanları şunlardır:

- type: Bütün ses kanalları STERO sabiti ile tanımlanacaktır.
- bufferSize: arabellekte tutulacak olan ses bilgileri 512 byte'lık parçalardan oluşacaktır.
- sampleRate: Bu değer bütün ses sinyalleri için 44100 olarak belirlenecektir.
- bitDepth: Ses sinyallerinin bit derinliği 16 olarak belirlenecektir.

Bu dört ses sinyal özelliği kullanıcı kontrollerinden bağımsız olarak başlangıç değerleri alacak ve bu değerler sistemin çalışması süresince değiştirilemeyecektir. Aracın çalışması boyunca, kullanıcı kontrolüne açık olan ses özellikleri ise şunlardır:

- Frekans
- Amplitude
- Pan
- Dalga formu
- Portamento

Bu ayrımlar ile, kullanıcının ses sinyalleri ile etkileşime girebileceği özellikler belirlenmiş olmaktadır. (Tablo 2.3)

Tablo 2.3 Ses özellikleri üzerindeki kullanıcı kontrolleri

Ses Özellikleri	Kullanıcı Kontrolü
type	-
bufferSize	-
sampleRate	-
bitDepth	-
Frekans	+
Amplitude	+
Pan	+
Dalga formu	+
Portamento	+

Bu aşamada kullanıcı kontrolüne imkan verilen ses özelliklerinden hangilerinin kamera ile eşlenebileceğine karar verilmelidir. Ses özelliklerinden frekans, amplitude ve pan kamera ile eşlenebilecek olan ses özellikleridir. Bu özellikler nitelikleri gereği, görüntülere eşlenerek aracımız için gerekli olan kamera kaynaklı seslerin üretilebilmesine yeterli olan değerleri oluşturabilecektir. Diğer ses özellikleri olan dalga formu ve portamento değerlerinin de kamera kaynaklı olması tarafımızca etkileşim açısından bir gereklilik olarak görülmemiştir. Bu değerler kullanıcı tarafından el ile belirlenecektir. Burada belirtilmesi gereken bir nokta, kamera kaynaklı olarak belirlenen ses değerlerinin, kullanıcı tarafından, dalga formu ve portamento değerleri gibi, el ile belirlenebilmesi imkanının da olacağıdır.

Bu durumda bir başka kullanıcı kontrolü kendisini gereklilik olarak dayatmaktadır. Kullanıcı kameradan alınan görüntüler ile belirlediği ses değerlerinin belli bir aralıkta durmasını isteyebilir. Kamera görüntülerinin, üzerinde tam kontrol sağlanamaz olan karakterinden dolayı, kameraya bağlanan ses özellikleri ile ilgili spektrum ayarlamalarının yapılabilmesi gerekmektedir. Bu yüzden kullanıcı, direk olarak ses değerlerini üretmeye katılmayan, ancak değerlerin nicelik aralıklarını belirleyen bu kontrollere sahip olacaktır. Spektrumun belirlenebilmesi için iki adet eşik değerinin bulunması gerekmektedir. Bunlar alt eşik değeri ve üst eşik değeridir. Dolayısıyla, kullanıcı üç adet kamera kaynaklı ses değerinin (frekans, amplitude, pan) spektrumlarını ayarlayabilmek için, alt ve üst eşik değerlerini belirleyeceği toplam altı adet kontrole sahip olacaktır. (Tablo 2.4)

Tablo 2.4 Kamera kaynaklı olan ve el ile belirlenebilen ses özellikleri

Ses Özellikleri	Kamera ile belirleme	El ile belirleme
Frekans	+	+
Frekans spektrumu	-	+
Amplitude	+	+
Amplitude spektrumu	-	+
Pan	+	+
Pan spektrumu	-	+
Dalga formu	-	+
Portamento	-	+

Bunların yanında tasarladığımız işitsel-sanat aracında, kameradan alınan görüntülerin hangi özelliklerinin nicelikleri ile oluşturulacağına karar verilmesi gerekmektedir.

Işık ortamında bütün renklerin oluşmasını sağlayan temel renkler olan kırmızı, yeşil, mavi ve bunların yanında dijital ortamda bütün renklerin sayısal karşılığı olan renk, doygunluk ve parlaklık değerleri, seslerin oluşturulmasında temel verileri sağlayacaktır. Toplamda, kamera görüntülerinin işlenmesi ile oluşturulacak olan değerlerin kaynağı şunlar olacaktır:

- Kırmızı
- Yeşil
- Mavi
- Renk
- Doygunluk
- Parlaklık

Kullanıcı, değerlerini kamera verileri ile oluşturabildiği ses özelliklerini (frekans, amplitude, pan), görüntü özelliklerinden istediklerini seçerek belirleyebilecektir. Ses

oluşumuna katılmasını istediği görüntü değerlerini seçebilecek ve istemediklerini devre dışı bırakabilecektir. (Tablo 2.5)

Tablo 2.5 Kullanıcının ses değerlerini istediği görüntü değerleri ile eşleyebilme imkanı

	Frekans	Amplitude	Pan
Kırmızı	+	+	+
Yeşil	+	+	+
Mavi	+	+	+
Renk	+	+	+
Doygunluk	+	+	+
Parlaklık	+	+	+

Kullanıcının ses değerlerini kamera görüntülerinden oluştururken sahip olduğu bir diğer kontrol, bu görüntü değerlerinin hangi oranda sesin oluşumuna katılacağı kararıdır. Kullanıcı, seçtiği görüntü değerlerinin hepsini aynı oranda sese katmak zorunda değildir. Bu yüzden, her bir görüntü özelliğinin, sesi oluşturan değerlere katılma oranının kullanıcı tarafından belirlenebilmesi gerekmektedir.

3. BULGULAR

3.1 ARACIN ÖZELLİKLERİ

3.1.1 Genel Özellikler

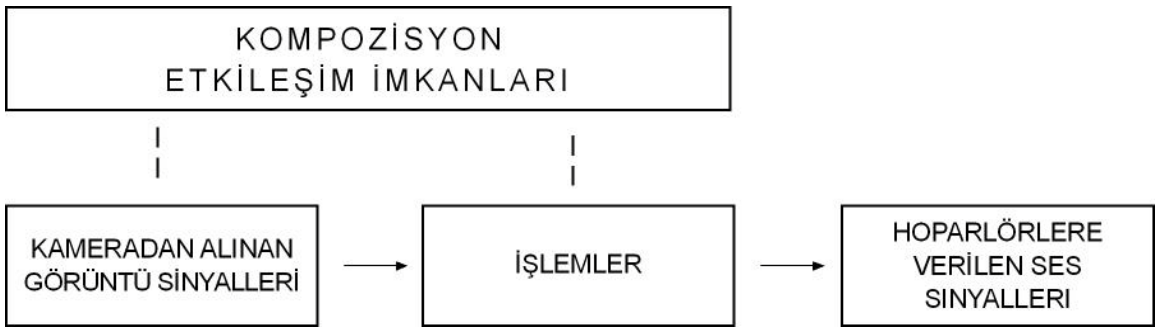
Bir önceki bölümde tartıştığımız nitelikler ve temel yönelimler ışığında, kamera kaynaklı işitsel-sanat aracı ve arayüzü somut düzlemde aşağıda belirtilen özelliklere sahip olacaktır.

Öncelikle, tasarlanan bu araç, kameranın işitsel-sanat enstrümanı olarak kullanılması amacının hayat bulmasını sağlayacak şekilde tasarlandı. Kullanıcı (işitsel-sanat icracısı) çıktı olarak elde ettiği ses sinyallerinin kaynağını kameradan almaktadır. Kameranın yakaladığı görüntülerin nicelik değerleri, ses sinyallerinin nicelik değerlerini gerçek zamanlı olarak belirlemektedir. Bu düzenek, kullanıcıya, kamerayı ses enstrümanı gibi kullanma imkanı sağlamıştır. (Şekil 3.1)

Şekil 3.1 Aracın temel işlevi

Ancak tasarladığımız aracın, kullanıcının kontrolüne izin verdiği tek etkileşim imkanı sadece kameranın gerçek zamanlı olarak yakaladığı görüntünün, yani işlenecek olan genel kadrajın, belirlenmesi değildir. Kullanıcı, görüntü ve ses sinyalleri ile ilgili olarak birçok özelliği değiştirebilme imkanına sahiptir. Bu değişiklik yapma imkanları sadece sistemin çalışmaya başladığı ilk anda değil, sistemin çalışmaya devam ettiği sürece devam etmektedir. Bu özellik, kullanıcıya düşündüğü kompozisyonun sonuçlarına göre ürettiği ses sinyallerinin yeniden düzenlenmesi imkanını tanımaktadır. Bu da, kullanıcının yeni denemelerini ses sinyallerinin üretim sürecinin içerisinde yapmasını sağlamakta ve tasarlanan araç ile kullanıcının etkileşim imkanlarını daha yüksek bir seviyeye çıkarmaktadır. Böylece kullanıcının yaptığı düzenlemeler ve aracın etkileşim

zenginliği, kameranın algılama sürecinden başlar ve bu görüntü sinyalleri işlemlerden geçirilirken hangi özelliklerin nasıl değerlendirileceğinin karar imkanlarıyla devam eder. (Şekil 3.2)



Şekil 3.2 Kullanıcı ile etkileşim imkanları

3.1.2 Görüntü ve Ses Sinyallerinin Tanımlayıcı Parametreleri

Araç, kameradan alınan görüntü sinyallerini belirli sayısal işlemlerden geçirerek bir takım parametreler çıkarmaktadır. Bu parametreler daha sonrasında ses sinyallerinin üretimi için gerekli olan sayısal verileri oluşturmaktadır. Ses sinyalleri için gerekli olan ve kullanıcının kamera çerçevesini her an yeniden belirleyerek sürekli olarak yeni parametrelerin yaratılmasını sağladığı görüntü özellikleri şunlardır:

- Kırmızı (red): Kamera kadrajı içerisinde yer alan piksellerin ortalama kırmızı değeridir. Bu değer 0 ile 255 arasında değişmektedir.
- Yeşil (yeşil): Kamera kadrajı içerisinde yer alan piksellerin ortalama yeşil değeridir. Bu değer 0 ile 255 arasında değişmektedir.
- Mavi (blue): Kamera kadrajı içerisinde yer alan piksellerin ortalama mavi değeridir. Bu değer 0 ile 255 arasında değişmektedir.
- Renk (hue): Kamera kadrajı içerisinde yer alan piksellerin ortalama renk değeridir. Bu değer 0 ile 255 arasında değişmektedir. Buradaki sayısal değerler, renk yelpazesi içerisindeki belli bir renge referans vermektedirler.
- Doygunluk (saturation): Kamera kadrajı içerisinde yer alan piksellerin ortalama doygunluk değeridir. Bu değer 0 ile 255 arasında değişmektedir.

- Parlaklık (brightness): Kamera kadrajı içerisinde yer alan piksellerin ortalama parlaklık değeridir. Bu değer 0 ile 255 arasında değişmektedir.

Renklerin fiziksel özellikleri ayrıntılı olarak daha önce çalışmanın 'Malzeme ve Yöntem' bölümünde incelenmiştir.

Aracın kameradan aldığı görüntülerin işlendikten sonra ürettiği parametrelerin ve kullanıcının kararlarının doğrultusunda belirlenen niceliklerin sayısal karşılık olarak, üretilen ses sinyallerinde belirleyeceği özelikler şunlardır:

- Frekans: Ses dalgasının 1 saniyedeki titreşim sayısıdır. Tasarladığımız araçta ses dalgalarının frekans aralığı 0 ile 10000 arasındadır. İnsan kulağının 20000 Hz frekansa kadar duyabilmesine rağmen, biz bu çalışmada üst sınırı 10000 Hz olarak belirledik. Bunun en temel nedeni, java platformunda kullandığımız ses sınıflarının çalışma performansları ile ilgilidir.
- Frekans spektrumu: Kullanıcının belirlediği ve üretilen ses dalgasının alt ve üst frekans aralığını oluşturan değerlerdir.
- Ses yüksekliği: Bu değer ses dalgasının genliği ile ilgilidir. 0 ile 1 arasında değer alır.
- Ses yüksekliği spektrumu: Kullanıcının belirlediği ve üretilen ses dalgasının alt ve üst genlik aralığını oluşturan değerlerdir.
- Pan: Ses dalgalarının çıkışta sağ ve sol ses kanalını kullanmasını sağlayan değerdir. -1 ile 1 arasında değer alır.
- Pan spektrumu: Kullanıcının belirlediği ve üretilen ses dalgasının pan değerinin alt ve üst aralığını oluşturan değerlerdir.
- Ses dalgası biçimi: Ses dalgası formunun karakterini veren seçimdir. Kullanıcı beş farklı ses dalgası seçeneğinden istediğini belirleyebilir. Bu seçenekler; PulseWave, SawWave, SineWave, SquareWave, TriangleWave'dir.
- Portamento: Ses dalgasının bir frekans değerinden başka bir frekans değerine geçişi sırasında oluşacak olan zaman aralığını belirler. Portamento değeri, 0 ile 2000 milisaniye aralığında yer alır.

Seslerin özellikleri ayrıntılı olarak daha önce çalışmanın ‘Malzeme ve Yöntem’ bölümünde incelenmiştir.

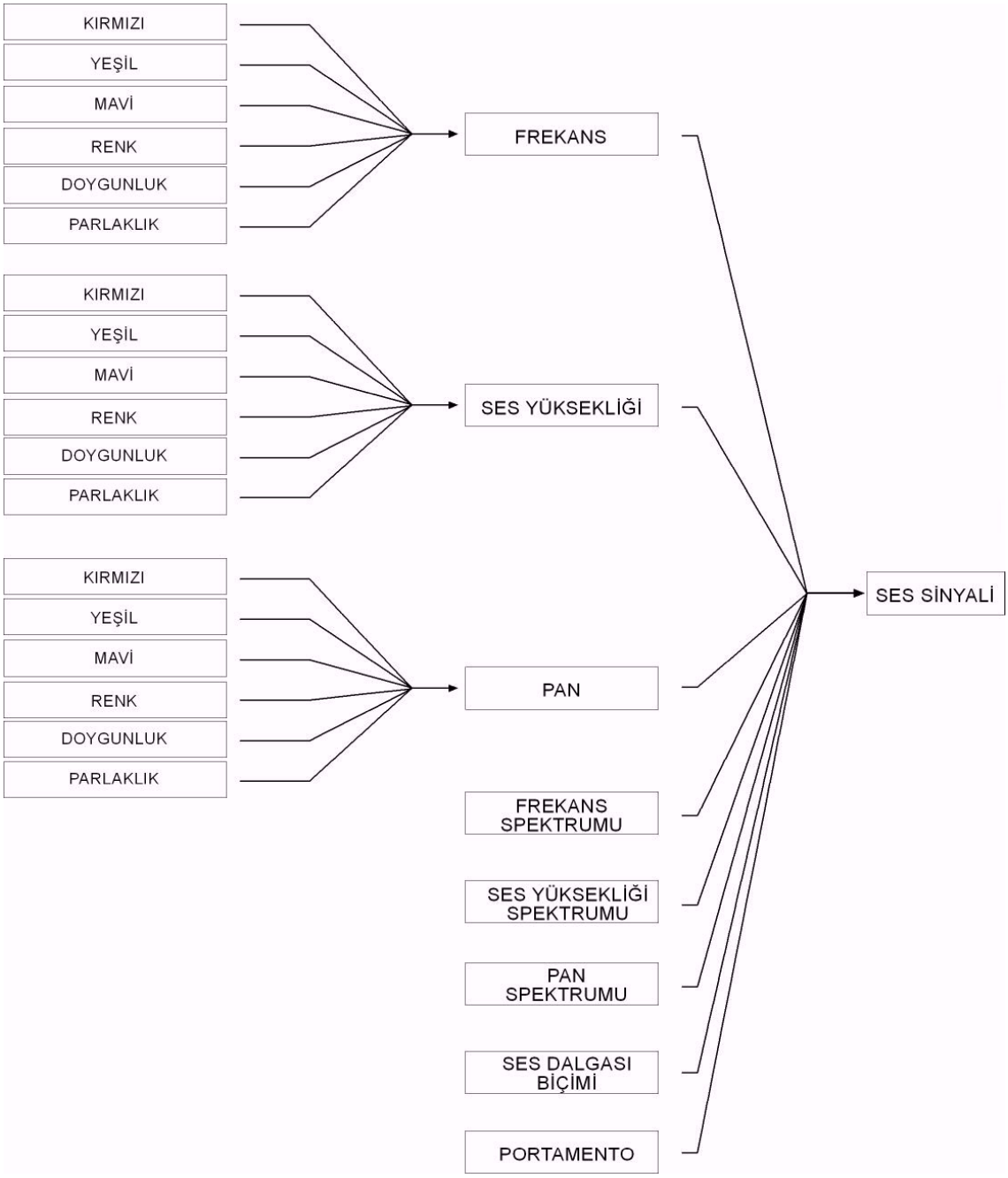
Kullanıcının görüntü ve ses ile ilgili olarak niceliklerini belirleyebildiği özellikler Şekil 3.3’te topluca gösterilmiştir. (Şekil 3.3)

görüntü	ses
KIRMIZI	FREKANS
YEŞİL	FREKANS SPEKTRUMU
MAVİ	SES YÜKSEKLİĞİ
RENK	SES YÜKSEKLİĞİ SPEKTRUMU
DOYGUNLUK	PAN
PARLAKLIK	PAN SPEKTRUMU
	SES DALGASI BİÇİMİ
	PORTAMENTO

Şekil 3.3 Kullanıcının niceliklerini belirleyebildiği görüntü ve ses özellikleri

3.1.3 Görüntü Sinyalleri ile Ses Sinyallerinin Eşlenmesi

Tasarladığımız araçta ses özelliklerinin hepsinin görüntü sinyallerine bağlanması işlevsellik açısından uygun görülmedi. Bu nedenle, ses özelliklerinden üç tanesinin (frekans, ses yüksekliği, pan) görüntü sinyalleri ile eşlenmesine karar verildi. Bu seçim yapılırken ses çıktısının etkisini zenginleştirecek ve kamera ile belirlenmesinin en anlamlı olacağı üç ses özelliği seçildi. Geri kalan diğer ses özellikleri (frekans spektrumu, ses dalgası türü, portamento) kullanıcının görüntü sinyallerinden bağımsız olarak belirlediği değerlerle oluşturuluyor. Buna göre kullanıcı görüntüden aldığı altı özelliğin (kırmızı, yeşil, mavi, renk, doygunluk, parlaklık) sayısal değerlerini ses sinyallerinin üç özelliği (frekans spektrumu, ses dalgası türü, portamento) ile istediği gibi eşleyebilecektir. (Şekil 3.4)



Şekil 3.4 Ses sinyalinin oluşturulmasında kullanıcının, kameradan gelen görüntünün değerleri ile ilgili olan ve kameradan bağımsız olan kontrolleri

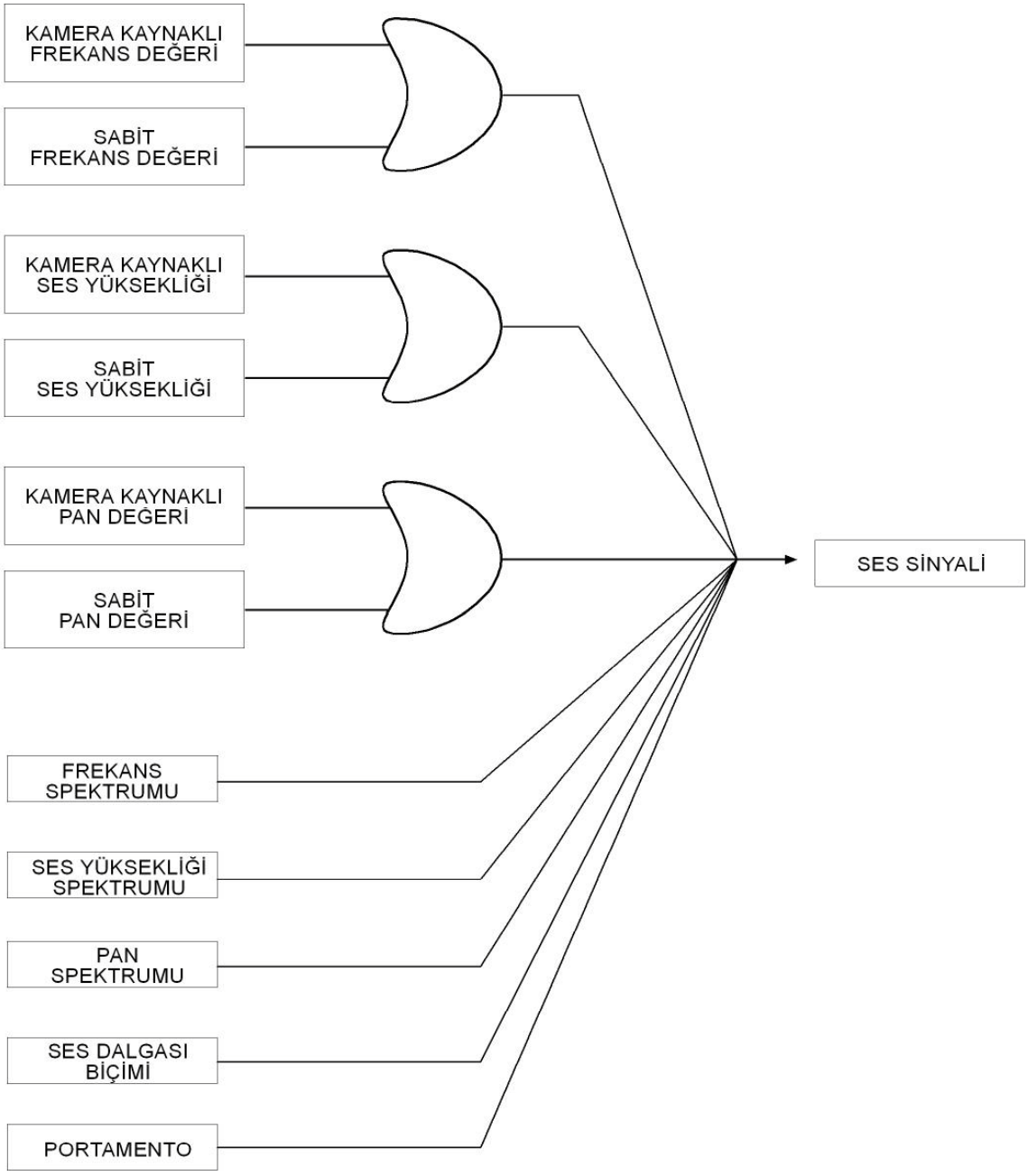
3.1.4 Ses Sinyallerinin Spektrum Kontrolleri

Tasarladığımız işitsel-sanat aracının önemli bir özelliği, yukarıda bahsedildiği ve Şekil 3.4'te görüldüğü üzere, spektrum kontrolleridir. Araçta, üç ses özelliğine bağlı spektrum kontrolleri vardır. Bu spektrum kontrolleri; ses sinyalinin frekans, yükseklik ve pan

değerini belli sınırlar dahilinde tutmaya yaramaktadır. Ses sinyallerinin bu özelliklerine spektrum koyma ihtiyacı, bu ses değerlerinin kameraya bağlı olmasından doğmaktadır. Sürekli değişen sinyaller kullanıcının kontrol edemediği bir sahaya geçebilir. Bunu engellemek; frekans spektrumu, ses yüksekliği spektrumu ve pan spektrumu ile mümkün olmaktadır. Ses sinyallerini oluşturan diğer özelliklere bu özelliğin eklenmesi gereksiz görüldü. Çünkü kamera verileri gibi değişken bir kaynağa sahip olmadıkları için portamento ve ses dalgası biçiminin değerlerinde spektrum kontrollerine ihtiyaç duyulmamaktadır.

3.1.5 Ses Kaynağının Kamera ya da Sabit Değer Olmasının Kontrolleri

Aracın tasarımında kullanıcıya tanınan bir başka kontrol, ses sinyallerinin üretiminde kullanılacak olan değerlerin (frekans, ses yüksekliği, pan) kamera kaynağı ile olan ilişkisini kesme yetkisidir. Burada amaçlanan şey, kullanıcıyı kameradan gelen değerlere mahkum olmasını engellemektir. Kullanıcı bu üç ses parametresini kendi kontrolü ile kamera değerlerine bağlayabilir ya da kamera ile olan ilişkilerini keserek kendi eliyle sabit değerler yaratabilir. Bu değerler için, kullanıcının arzusu ile yine kameraya dönüş yapılmadan kendi insiyatifi ile farklı sayısal nicelikler belirlenmeye devam edilebilir. (Şekil 3.5)

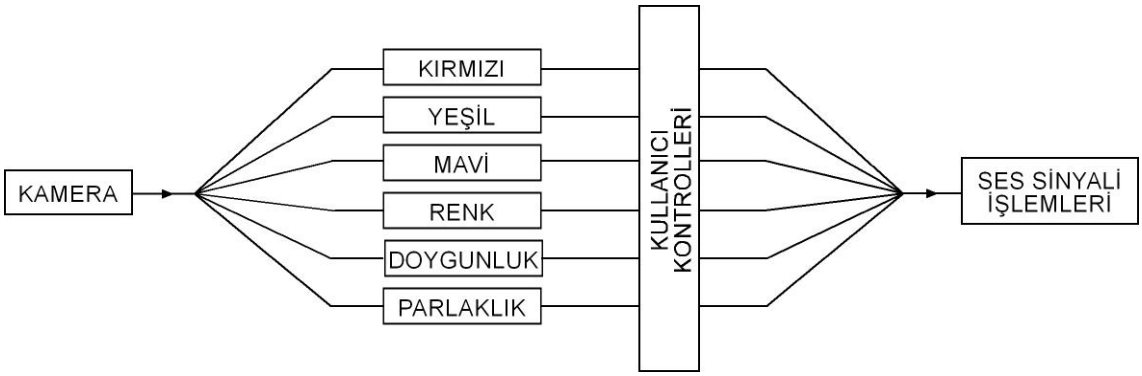


Şekil 3.5 Kullanıcının kamera kaynaklı değer üretme karar kontrolü

3.1.6 Kamera Kaynağından Alınan Görüntü Parametrelerinin Ses Sinyallerine Katılımında Oranlarının Belirlenmesinin Kontrolü

İşitsel-sanat aracının kameradan aldığı görsel sinyallerin sayısal parametrelere çevrilmesi, kullanıcıyı başka bir kontrol ile baş başa bırakır. Bu kontrol, hangi görüntü

parametresinin ses sinyalini hangi oranda etkileyeceğinin belirlendiği kontroldür. Bu kontrol, işitsel-sanat icracısına görüntü sinyalleri arasında farklı ilişkiler yaratarak değişik ses denemeleri yapma imkanı tanır. Görüntü sinyallerinden oluşturulan altı farklı sayısal parametre (kırmızı, yeşil, mavi, renk, doygunluk, parlaklık) kullanıcının isteklerine göre farklı ağırlıklarda ses değerlerine katılabilir. İster bütün görüntü parametreleri, ister tek bir tanesi ya da bir kaçı ses sinyali yaratma sürecine katılabilir. (Şekil 3.6)



Şekil 3.6 Kullanıcının görüntü değerlerini kendi kontrolünden sonra ses sinyallerinin üretileceği işlemlere geçirmesi

3.1.7 Kamera Çerçevesinin Alt Çerçevelere Bölünmesi

3.1.7.1 Kamera çerçevesinde alt çerçeveler oluşturulmasının genel mantığı

Tasarlanan işitsel-sanat aracının, kullanıcı için ses sinyallerinin oluşturulmasında yarattığı zengin etkileşim imkanlarının, ses sinyallerine kaynak teşkil eden görüntü sinyallerinin oluşturulduğu kamera çerçevesinde de yaratılması gerektiği düşünüldü. Bu amaçla, kamera ekranının alt çerçevelere bölünmesine karar verildi. Görüntü kaynağının alt çerçevelere ayrılmasının tasarlanmasında, iki önemli çıkış noktası vardır.

Bunlardan ilki; kullanıcıya, ses sinyallerine kaynaklık eden görüntülerin kontrolü için daha geniş bir imkan sağlamaktır. Böylece kullanıcı görüntü kaynağı olarak bütün kamera çerçevesini kullanmak zorunda kalmamaktadır. Bunun yerine kendi alt çerçevelerini oluşturarak sadece bunların görüntü kaynağı olarak değerlendirilmesini

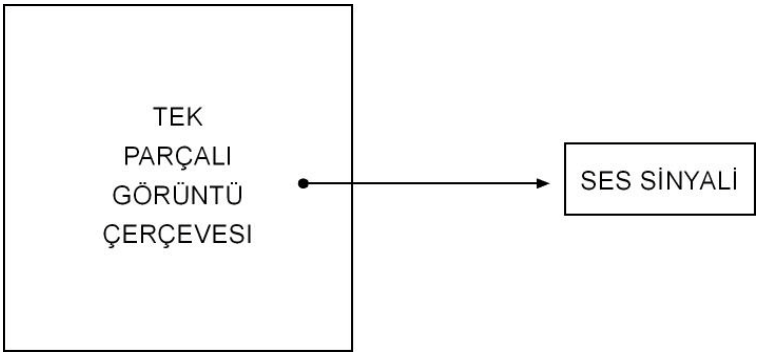
sağlayabilir. Bu alt çerçevelerden istediğini aktif hale ve istediklerini pasif konuma getirebilir. Böylece kullanıcı, aracın bu imkanı ile yakalanan kamera görüntüsü üzerinde de kontrol sahibi olmaktadır.

İkinci çıkış noktası; her ses sinyalinin belli bir alanda oluşan görüntü sinyallerinden üretildiği düşünülürse tek bir kameradan alınan görüntünün alt parçalara ayrılması ile daha çok sayıda ses sinyali için veri elde edilmiş olur. Böylece artık kameranın çerçevesinden elde edilecek tek bir parametreler dizisi değil, alt çerçeve sayısı kadar parametreler dizisi elde edilir. Bu sayede de, farklı görüntüler için farklı kameraların sisteme bağlanması gibi hem donanım hem de yazılım olarak sisteme külfet getirecek bir yöntem yerine, tek bir kamera görüntüsünün parçalanması ile yine birden çok kaynağa ulaşılmış aynı zamanda da donanım ve yazılım olarak sistem zorlanmamıştır.

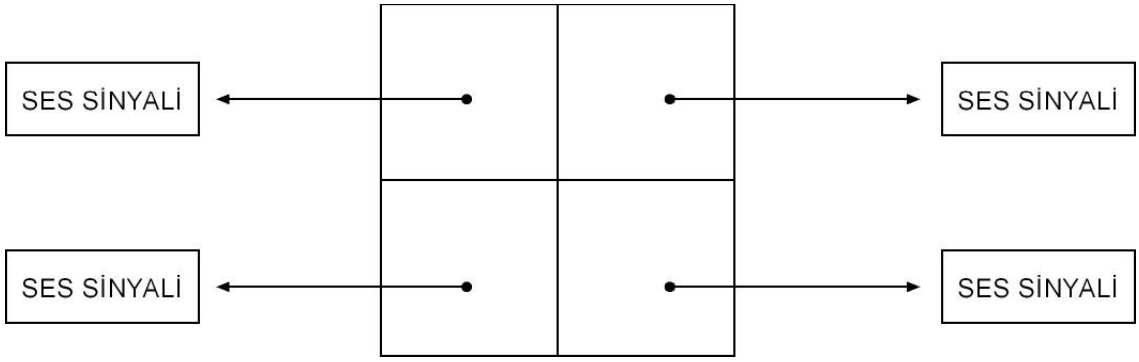
Bu bakış açısı ile, kullanıcıya kamera çerçevesini dört ve on altı alt çerçeveye bölme imkanları tanınmaktadır. Yukarıda anlatılan ve görüntü sinyalleri ile ses sinyalleri arasındaki etkileşimi gösteren kısımlar sadece bir ses kanalının yaratılması ile ilgilidir. Kamera kaynağının alt çerçevelere bölünmesi ile aynı süreç daha fazla ses kanalı için gerçekleşecektir. Bu şekilde arttırılan ses kanallarının çalışma prensipleri de aynı şekilde olacaktır.

3.1.7.2 Kamera çerçevesinin dört alt çerçeveye bölünmesi

Kamera kaynağının dört alt parçaya bölünmesi seçeneği ile, oluşturulan ses kanalları dörde çıkarılmaktadır. Bu şekilde sistem, görüntüdeki her çerçeveyi farklı bir veri kaynağı olarak değerlendirerek, bu kaynakların farklı niceliklere sahip dört ayrı değerler dizisine sahip olduğunu düşünerek dört ayrı ses kanalı yaratır. Kullanıcı, bu ses kanallarının hem farklı görüntü kaynaklarından oluşturulması, hem de her bir ses kanalı için farklı değerler belirleyebilmesi ile istediği ses mimarisini yaratabilecektir. (Şekil 3.7 ve Şekil 3.8)



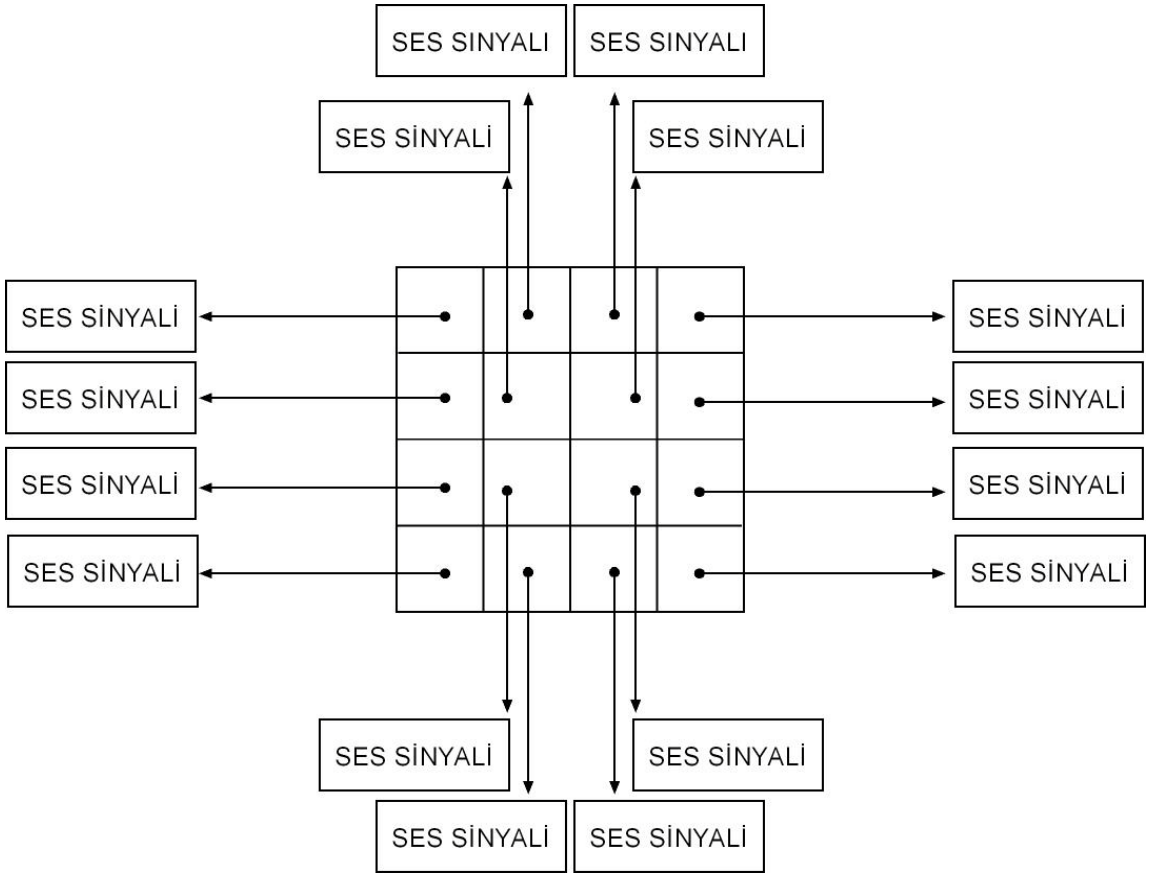
Şekil 3.7 Tek bir görüntü çerçevesinden bir ses kanalı elde edilmesi



Şekil 3.8 Dört farklı görüntü çerçevesinden dört farklı ses kanalı elde edilmesi

3.1.7.3 Kamera çerçevesinin on altı alt çerçeveye bölünmesi

Kamera kaynağının on altı alt parçaya bölünmesi seçeneği ile, oluşturulan ses kanalları on altıya çıkarılmaktadır. Bu çerçevelerde oluşan görüntü kaynaklarının farklı niceliklere sahip on altı ayrı değerler dizisine sahip olduğu düşünülerek, on altı ayrı ses kanalı yaratılır. Kullanıcı, bu ses kanallarının hem farklı görüntü kaynaklarından oluşturulması, hem de her bir ses kanalı için farklı değerler belirleyebilmesi ile istediği ses mimarisini yaratabilecektir. (Şekil 3.9)

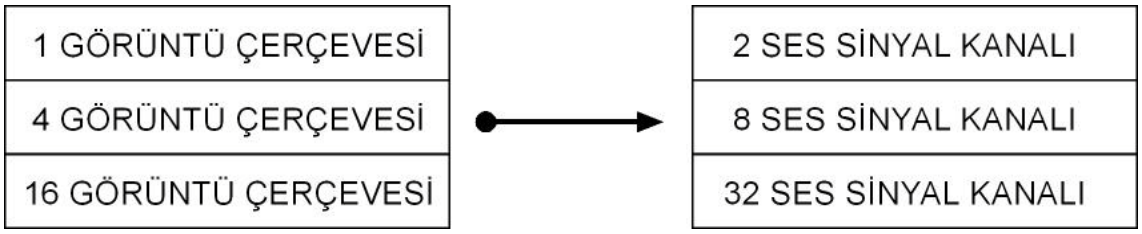


Şekil 3.9 On altı farklı görüntü çerçevesinden on altı farklı ses kanalı elde edilmesi

3.1.8 Her Bir Görüntü Çerçevesinden İki Ayrı Ses Sinyal Kanalı Üretilmesi

Tasarlanan işitsel-sanat aracının her bir görüntü çerçevesinden bir ses kanalının yaratılması ve böylece kullanıcıya, tek çerçeveli, dört çerçeveli ve on altı çerçeveli düzenlemeler yapma imkanlarının verilmesi aracın zengin ve farklı ses sinyalleri üretmesini sağlamaktadır. Ancak bu noktada, bir etkileşim sorunu ile karşı karşıya gelinir. Kullanıcının aynı görüntü sinyallerinden farklı özelliklerde ses sinyalleri oluşturma isteği de cevaplanmalıdır. Çünkü aynı çerçeveden gelen görüntü sinyalleri ile her türlü ses özelliklerine sahip farklı sinyaller üretilebilir. Bunun yanında, kullanıcı en temel olarak, ses çıkışlarında sağ ve sol hoparlör denetimlerini kendisi yapmak isteyebilir. Bu sebeplerden dolayı, her görüntü çerçevesi için, kullanıcıya iki ses sinyali yaratma imkanı sağlanmaktadır. Bu sayede, aynı görüntü çerçevesinden iki ayrı özelliklere sahip ses sinyalleri yaratılabilmektedir.

Bu özellikler kullanıcının üretebileceği ses sinyal kanallarının iki kat artmasını sağlamaktadır. Böylece kullanıcı tek çerçeveli bir görüntü kaynağı ile çalışırken iki, dört çerçeveli görüntü kaynağı ile çalışırken sekiz, on altı çerçeveli görüntü kaynağı ile çalışırken otuz iki ses kanalı yaratabilmektedir. Bu sayıya ulaşabilmiş bir ses yelpazesinin, oldukça zengin ses düzenlemeleri için yeterli olabileceği düşünülmüştür. (Şekil 3.10)



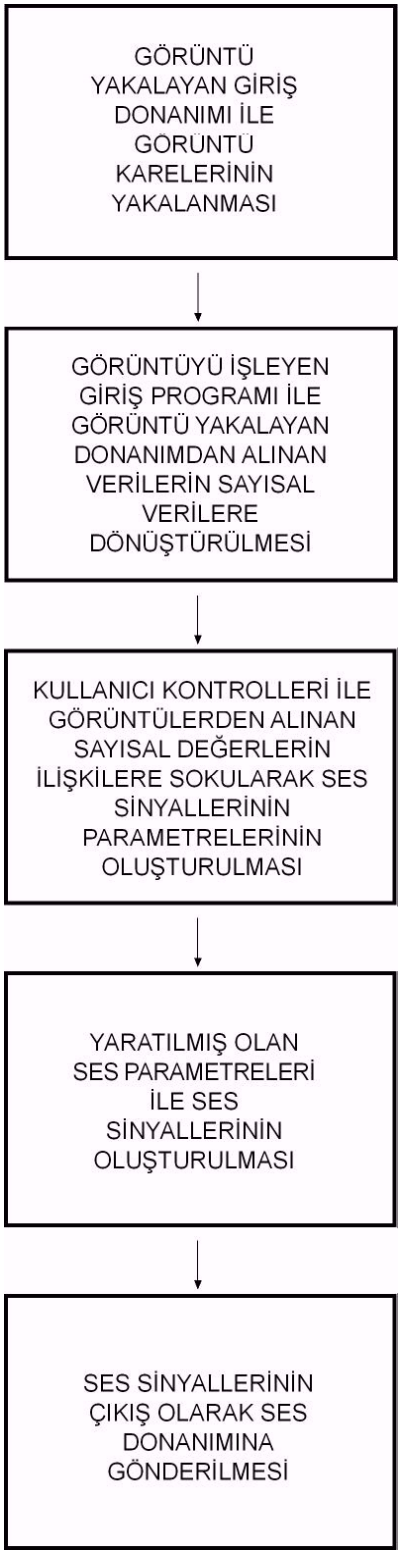
Şekil 3.10 Her bir görüntü çerçevesinden iki ayrı ses sinyal kanalı elde edilmesi

3.2 SİSTEMİN YAPISI

3.2.1 Genel Yapı

Tasarlanan aracın üzerine oturduğu teknik yapı, işlevsel düzeneğe paralel olarak oluşturuldu. Aracın işlevsel yapısı yani kullanıcının hedef ve etkileşimlerinin düzenlenmesi; kullanıcının kamera ile yakaladığı görüntü çerçevelerinden gerçek-zamanlı olarak sayısal parametreler oluşturmak ve bunların yine kullanıcı inisiyatifi ile belirlenen ilişkiler ile ses sinyallerinin özelliklerini belirlemesi üzerine oturtuldu. (Şekil 3.1 ve Şekil 3.2)

Aracın teknik sistemi de, etkileşim düzeneğine paralel olarak işlevsel parçacıkların arka arkaya getirilmesi ile meydana gelmektedir. Aracın yapısı en genel olarak; görüntü yakalayacak olan donanımdan dijital verilerin alınması, bu verilerin işlenerek anlamlı sayılara dönüştürülmesi, bu sayıların kullanıcı kontrolü ile ilişkilere sokularak ve ek ses özelliklerinin katılımı ile ses sinyallerini oluşturacak olan parametrelerin yaratılması ve son olarak buradan çıkan nicelikler ile ses sinyallerinin yaratılması ve hoparlörlere yansıtılmasından oluşmaktadır. (Şekil 3.11)



Şekil 3.11 Aracın genel yapısı

Şekil 3.11'de görüldüğü gibi; aracın çalışma sistemi, birbirini takip eden ve birinin çıkış parametrelerinin bir diğerinin giriş parametrelerini oluşturduğu art arda gelen modüllerden oluşmaktadır. Bu şekilde oluşturulmuş olan yapı, programın sağlamlığı açısından önemli olmaktadır. Bunun yanında, programdaki karmaşıklığın azaltılması ve sade bir yazılım teknolojisinin gereği olarak modüler bir programlama dili gereklidir. Bu özellikler hem üretim aşamasında hem de üretim sonrasında bakım ve geliştirme çalışmaları için gereklidir. Daha sonrasında aracın geliştirilmesi ve yeni özelliklerin eklenmesi, programlama yapısının modüler ve sade tasarımı sayesinde daha rahat olacaktır.

3.2.2 Sistem Yapısını Oluşturan Modüller ve Verilerin Dönüştürülmesi

3.2.2.1 Görüntüyü işleyen giriş modülü

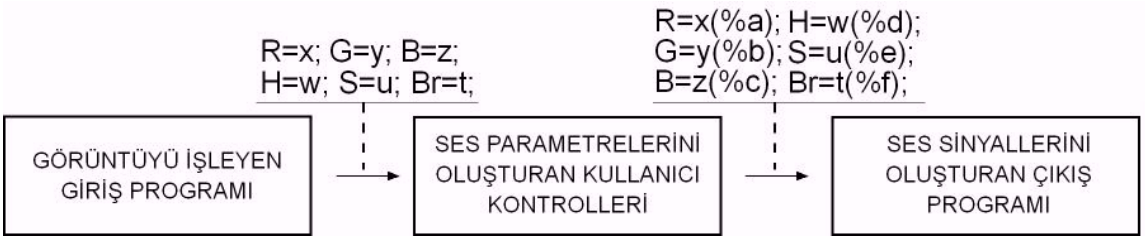
Sistemin bu kısmında kullanılan giriş programının amacı, görüntü algılayıcı donanımdan alınan görüntülerin, araç işleyişi için gerekli olan anlamlı sayısal değerlere dönüştürülmesidir.

Bu giriş programında; dijital veriler alınmakta ve görüntü işleme yöntemiyle, kullanıcının kontrolüne sunulmak üzere kırmızı, yeşil, mavi, renk, doygunluk ve parlaklık değerleri çıkış parametreleri olarak oluşturulmaktadır. Burada oluşan çıkış parametreleri modüler sistemde bir sonraki adımda bulunan, kullanıcı kontrolü ile ses sinyal değerlerini oluşturacak olan program için giriş parametreleri olacaktır. (Şekil 3.12)

Şekil 3.12 Görüntü işleyen giriş modülünün giriş ve çıkış değerleri

3.2.2.2 Görüntü parametrelerini ses sinyal parametrelerine dönüştüren modül

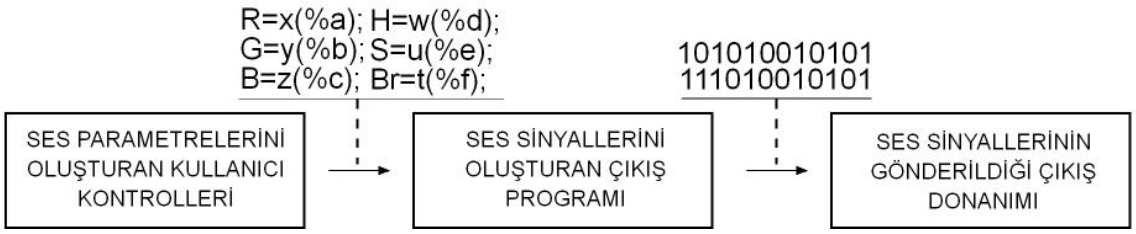
Bir önceki alt başlıkta işlediğimiz giriş modülünün çıkış değerleri, kameradan alınan görüntülerin işlenmesiyle oluşturulan kırmızı, yeşil, mavi, renk, doygunluk ve parlaklık değerleridir. Bu alt başlıkta incelediğimiz modülün giriş değerlerini ise, giriş modülünün çıkış değerleri oluşturur. Burada amaç, görüntü değerlerinin kullanıcının kontrolü ile anlamlı ses parametrelerine dönüşmesini sağlamaktır. Giriş modülünden alınan görüntü değerleri, kullanıcının yeni düzenlemeleri ile yüzde değerleri alacaktır. Ses sinyallerinin değerleri de, görüntüye bağlanmış ve görüntüye bağlanmamış olan niceliklerin düzenlenmesi ile oluşturulur. Sonuç olarak bu modül, giriş olarak aldığı görüntü değerlerini, ses sinyallerini yaratacak olan modüle çıkış olarak vermek üzere, gereken bütün ses parametrelerini elde edecek şekilde işler. (Şekil 3.13)



Şekil 3.13 Kullanıcı kontrollerini oluşturan modülün giriş ve çıkış değerleri

3.2.2.3 Ses sinyal parametrelerinden ses sinyallerini oluşturan modül

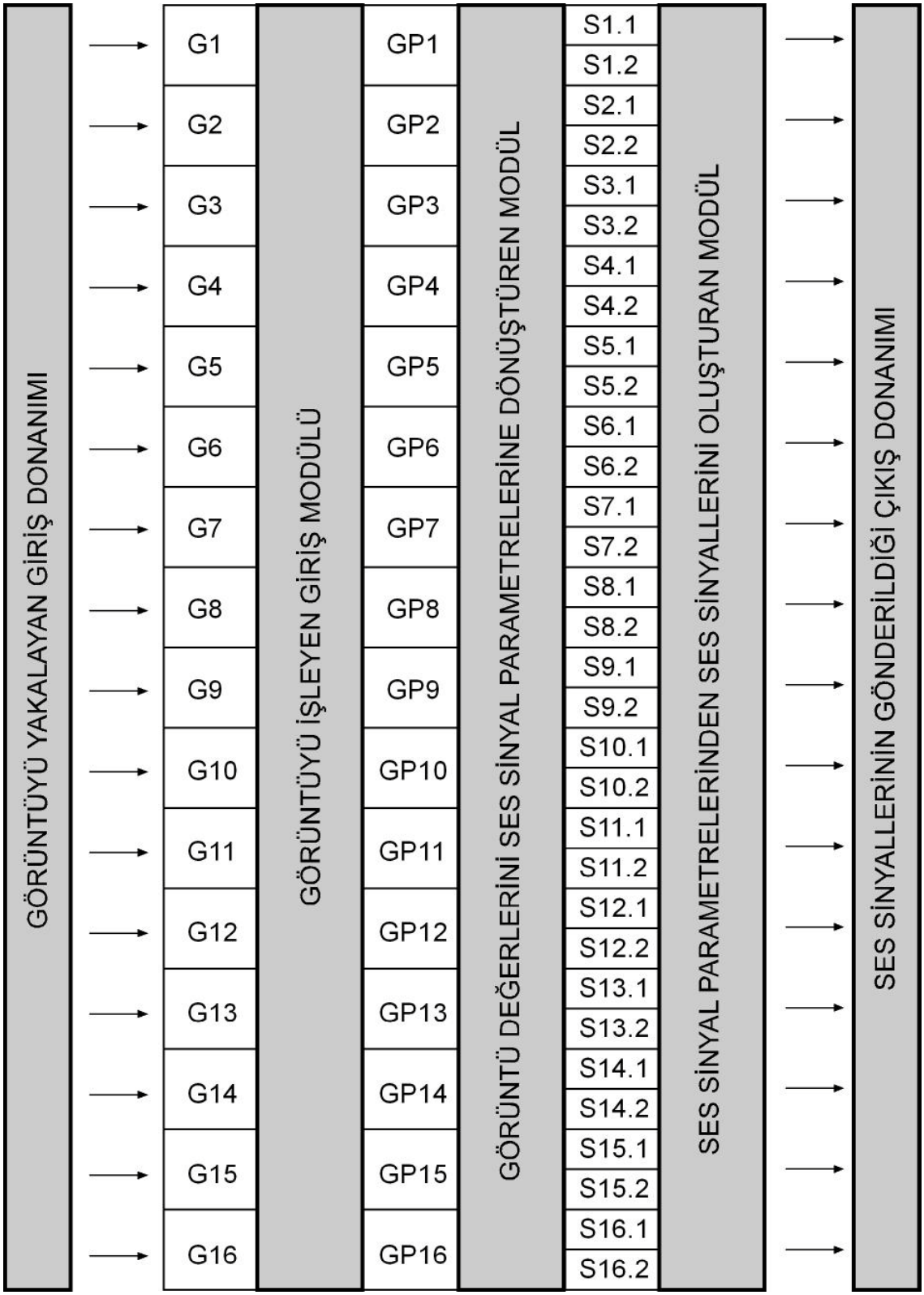
Görüntü değerlerini kullanıcı kontrolleri ile ses parametrelerine dönüştüren modül, çıkış değerleri olarak bu parametreleri, ses sinyallerini oluşturan modüle verir. Sinyalleri oluşturan modül, ses parametrelerini giriş değeri olarak alır ve çıkış olarak ses sinyallerinin kendilerini oluşturarak aracın bağlı bulunduğu ses donanımına verir. Giriş donanımından alınan görüntü ile başlayan süreç, bu modülün ses sinyallerini üretmesi ve çıkış donanımına göndermesi ile son bulur. (Şekil 3.14)



Şekil 3.14 Ses sinyallerini oluşturan ve çıkış donanımına gönderen modül

3.2.3 Paralel Çalışan Görüntü ve Ses Kanalları

Yukarıda anlatılan süreç, sadece tek bir görüntü ve ses kanalının geçirdiği süreçtir. Görüntü yakalayan giriş donanımından, ses sinyallerinin gönderildiği çıkış donanımına kadar çalışan bütün modüller sadece bir kanal için anlatılmıştır. Çalışmanın 'Aracın Özellikleri' kısmında anlatılan özellikler, kullanıcı kontrolüne bağlı olarak bir, dört ve on altı görüntü çerçevesinden sırasıyla iki, sekiz ve otuz iki kanal ses elde edilmesi hakkında aracın sahip olduğu yetiyi açıklamaktadır. Bu durum, araçta paralel olarak işleyen ve sürekli olarak dönüştürülen verilerin, yine paralellik içerisinde taşınmasını da gerektirmektedir. (Şekil 3.15)



Şekil 3.15 Kullanıcı kontrolü ile otuz iki ses kanalına çıkabilecek veri akışı
(Gx, görüntü kanalı; GPx, görüntü parametresi; Sx.1 ve Sx.2 ses parametresi)

3.3 SİSTEMİN ÇALIŞMASI

Burada, önceki kısımlarda genel özellikleri ve sistem yapısı incelenmiş olan kamera kaynaklı işitsel-sanat aracının yazılım olarak algoritmaları, gereksinim özellikleri ve kod parçaları incelenerek sisteminin çalışma şekli ortaya konulacaktır.

3.3.1 Genel İşleyiş

Araç, processing ortamında java programlama dili kullanılarak yaratıldı. Processing ortamının en temel karakteri modüler çalışma yapısıdır. Java dilinde bulunan temel karakterler, processing ortamı ile sağlam bir şekilde bütünleşmiş durumdadır.

Java dilinde bulunan init( ) fonksiyonu, processing dilinde setup( ) fonksiyonu ile karşılanmaktadır. Bu fonksiyon program çalışmaya başladıktan sonra sadece bir kere çalıştırılır ve çalışma süresi boyunca tekrar bu fonksiyona geri dönülmez.

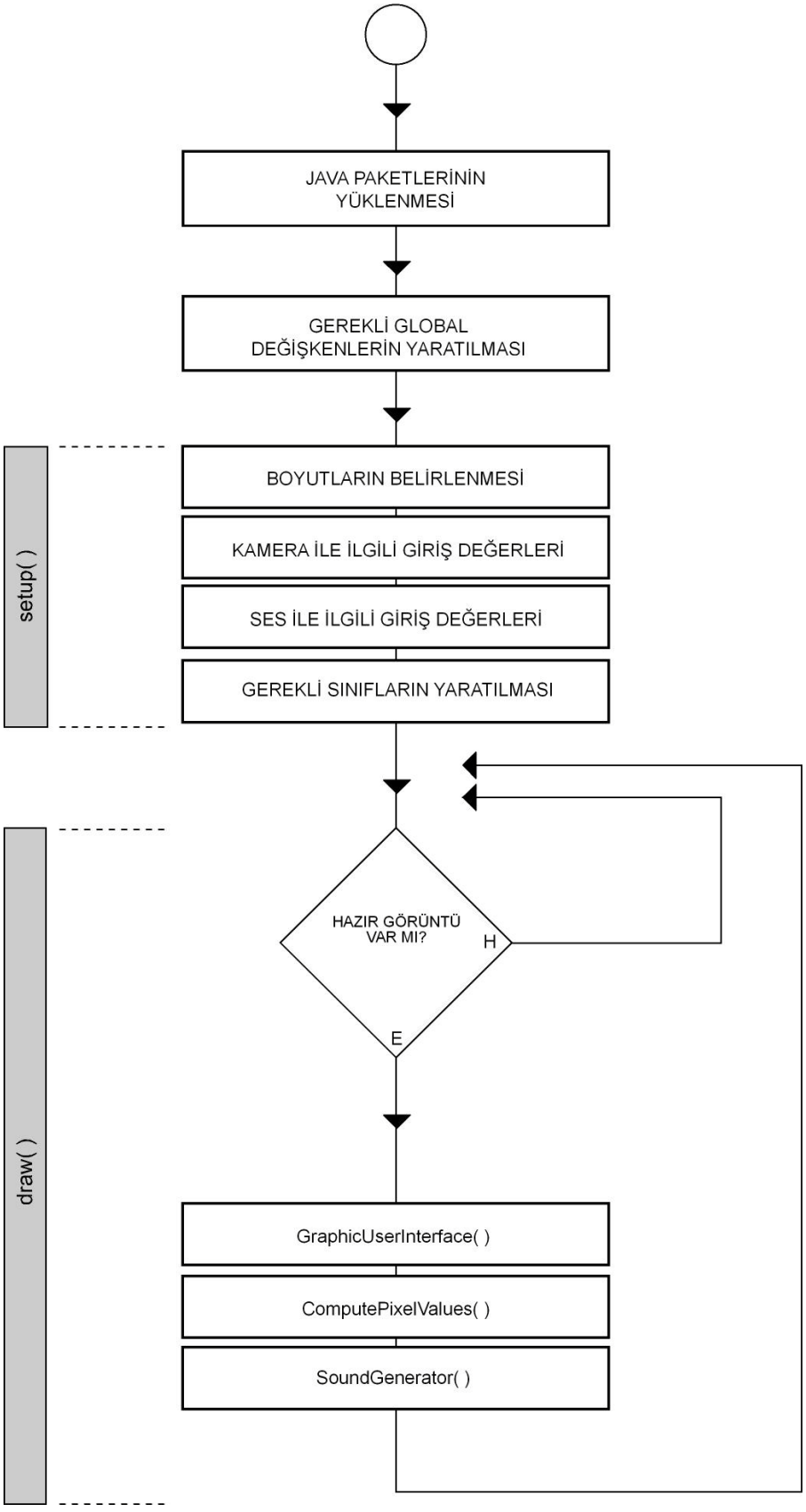
Processing ortamının java diline kattığı en temel özelliklerden biri de, draw( ) fonksiyonudur. Bu fonksiyon, program çalışmaya başladıktan ve setup( ) fonksiyonu bir kere çalıştırıldıktan sonra sürekli olarak kendisini baştan çağırır. Bu döngü fonksiyonu, program işleyişinde herhangi bir kesilme olmaz ise sonsuza kadar tekrar eder. Processing ortamında üretilen yazılımların temeli draw( ) fonksiyonu ile oluşturur.

Bizim de çalışmamızda temel iskelet, setup( ) ve draw( ) fonksiyonları üzerine oturtuldu. Giriş değerlerinin ayarlanması için setup( ) fonksiyonu kullanıldı. Bu fonksiyon içerisinde kullanıcının etkileşime gireceği alanın boyutlarının belirlenmesi, kameranın ve kamera ile ilgili sınıfının giriş değerlerinin ayarlanması, ses sinyallerinin giriş değerlerinin ayarlanması ve program için gerekli olan sınıfların yaratılması işlemleri gerçekleşmektedir.

Giriş değerlerinin oluşturulmasının ardından, ürettiğimiz yazılımın gövdesini oluşturan draw( ) fonksiyonu, öncelikli olarak kameradan yakalanmış hazır bir görüntünün olup olmadığını sınamaktadır. Eğer işlenmek üzere ulaşılabilir bir görüntü hazır değil ise;

sürekli olarak kendi koşullarını tekrar sınamaya devam eder. Ulaşılabilir bir görüntü hazır olduğunda ise; sırasıyla GraphicUserInterface( ), ComputePixelValues( ), SoundGenerator( ) fonksiyonlarını çalıştırır. Bu fonksiyonlardan ilki, kullanıcının etkileşime girdiği arayüzün oluşturulması ve etkileşim sonucunda yeni değerlerin hesaplanması işlemi ile ilgilenir. Böylece arka planda çalışan sistem ile kullanıcının ilişkiye sokulmasının denetimleri gerçekleştirilmiş olur. İkinci fonksiyon, kameradan alınan görüntülerin çalışmanın önceki kısımlarında anlatılan ilkeler doğrultusunda işlenmesini ve ses için anlamlı değerlerin üretilmesini sağlar. Üçüncü fonksiyon, ses için oluşturulmuş olan son parametrelere göre ses sinyallerinin oluşturulmasını sağlar.

Bu işlemlerin gerçekleştirilmesi, draw( ) fonksiyonu içerisinde sürekli olarak sonsuz döngü şeklinde devam eder. Böylece gerçek zamanlı görüntü değerleri ile ses sinyallerinin oluşturulmasını sağlayan araç işler hale gelir. (Şekil 3.16)



Şekil 3.16 Aracın işleyişinin akış diyagramı

3.3.2 Başlangıç Değerlerinin Yaratılması ve Genel Ayarlamaların Yapılması

3.3.2.1 Gerekli paketlerin yüklenmesi

Başlangıç olarak, yazılımın ihtiyaç duyulan sınıfları, arayüzleri ve fonksiyonları kullanabilmesini sağlamak amacıyla gerekli olan paketlerin yüklenmesi gerekmektedir. Burada ihtiyaç duyulan üç adet paket vardır:

- processing.video:

  Bu paket, görüntü yakalama aygıtı olarak kullanılan kameradan görüntü karelerinin alınmasını sağlamak için gereklidir. Ayrıca işlenecek olan görüntüler bu paketin altındaki Capture sınıfından üretilmiş olan nesnelerinin bir özelliği olarak bulanacaktır.
- ddf.minim:

  Bu paket, ses kanallarının oluşturulması ve oluşturulan ses sinyallerinin kanallara verilmesi için gerekli olan sınıf ve arayüzleri içerir. Ayrıca ddf.minim paketinin çalıştırılması için gerekli olan başlangıç komutu da bu paket içerisindeki Minim sınıfında mevcuttur.
- ddf.minim.signals:

  Bu paket, ses sinyallerin üretilmesi için gerekli olan sinyal sınıflarına ulaşabilmek için gereklidir.

3.3.2.2 Gerekli global değişkenlerin tanımlanması

Ürettiğimiz yazılım için gerekli olan paketler yüklendikten sonra, programın bir çok yerinde kullanılacak olan global değişkenler tanımlanmıştır. Bu değişkenler yazılımın çalışması boyunca faklı işlevler taşıyacaktır. Bu global değişkenler şunlardır:

- cam: Bu değişken, processing.video kütüphanesinin standart bir sınıfı olan Capture sınıfından üretilmiş olan bir nesneyi tutmakla görevli Capture türünde bir değişkendir. Görüntü yakalama cihazının içeriği ile Capture türündeki bu değişken üzerinden ilişki kurmaktayız.
- outs[ ][ ]: df.minim paketinin standart bir sınıfı olan AudioOutput sınıfından üretilmiş olan nesneleri tutmakla görevli iki boyutlu bir dizi değişkenidir.

- oscs[ ][ ]: ddf.minim.signals paketinin standart bir sınıfı olan Oscillator sınıfından üretilmiş olan nesneleri tutmakla görevli iki boyutlu bir dizi değişkenidir.
- synthStore[ ][ ]: Bir ses sinyali için gerekli olan bütün parametreleri içerisinde taşıması için tanımladığımız Synth sınıfından üretilmiş olan bütün nesneleri tutmakla görevli iki boyutlu bir dizi değişkenidir.
- activeButtons[ ]: Ses sinyallerine aktarılacak olan görüntü parametrelerinin hangileri olduğunun belirlenmesi için tanımladığımız ActiveButton sınıfından üretilmiş olan bütün nesneleri tutmakla görevli tek boyutlu bir dizi değişkenidir.
- activeButtonHandles[ ]: Ses sinyallerine aktarılacak olan görüntü parametrelerinin oranlarının belirlenmesi için tanımladığımız ActiveButtonHandle sınıfından üretilmiş olan bütün nesneleri tutmakla görevli tek boyutlu bir dizi değişkenidir.
- monitorings[ ]: Ses sinyallerine aktarılacak olan görüntü parametrelerinin hangilerinin olduğu ve hangi oranda katılacağı belirlendikten sonra, son değerlerinin tutulmasını sağlamak amacı ile tanımladığımız Monitoring sınıfından üretilmiş olan bütün nesneleri tutmakla görevli tek boyutlu bir dizi değişkenidir.
- monitoringManuals[ ]: Ses sinyallerinin ilgili değerlerinin kamera tabanlı ya da el ile belirlenmesinin bilgisini taşıması için yaratmış olduğumuz MonitoringManual sınıfından üretilmiş olan bütün nesneleri tutmakla görevli tek boyutlu bir dizi değişkenidir.
- spectrums[ ]: Monitoring sınıfının nesnelerinin değerlerinin hangi aralıkta olacağının belirlenmesi için yaratmış olduğumuz Spectrum sınıfından üretilmiş olan bütün nesneleri tutmakla görevli tek boyutlu bir dizi değişkenidir.
- portamento: Ses sinyallerinin portamento değerlerinin ne olacağı ile ilgili olarak yarattığımız PortamentoTool sınıfından üretilmiş olan bütün nesneleri tutmakla görevli bir değişkenidir.
- portabutton: Ses sinyallerinin portamento değerlerinin açık olup olmayacağı bilgisini tutan ve arayüzün işleyişi gereği LeftSideButton adında tanımladığımız sınıftan üretilmiş bir nesneyi tutan değişkendir.

- previewbutton: Kullanıcının kamera ile yakalanmış olan görüntüleri izleyip izlememe isteğinin bilgisini tutan ve arayüzün işleyişi gereği LeftSideButton adında tanımladığımız sınıftan üretilmiş bir nesneyi tutan değişkendir.
- gridbutton: Kullanıcının kamera ile yakalanmış olan görüntüleri ızgaralı bir şekilde izleyip izlememe isteğinin bilgisini tutan ve arayüzün işleyişi gereği LeftSideButton adında tanımladığımız sınıftan üretilmiş bir nesneyi tutan değişkendir.
- linebutton: Kullanıcının görüntü parametrelerinin ses sinyal parametreleri ile olan ilgilerini net bir şekilde görsel olarak sunmak amaçlı çizgilerin görülüp görülmeyeceği bilgisini tutan ve arayüzün işleyişi gereği LeftSideButton adında tanımladığımız sınıftan üretilmiş bir nesneyi tutan değişkendir.
- gridNumberSelections[ ]: Kameradan gelen görüntünün kaç parçalı (bir, dört ya da on altı) olacağının bilgisini tutan ve arayüzün işleyişi gereği LeftSideButton adında tanımladığımız sınıftan üretilmiş nesneleri tutan tek boyutlu bir dizi değişkendir.
- signalNumberSelections[ ]: Kameradan alınan her görüntü çerçevesinden üretilebilen iki ses sinyal kanalından hangisinin aktif seçim halinde olduğunun bilisini tutan ve arayüzün işleyişi gereği LeftSideButton adında tanımladığımız sınıftan üretilmiş nesneleri tutan tek boyutlu bir dizi değişkendir.
- signalKindSelections[ ]: Üretilen ses sinyalinin beş sinyal türünden hangisi olacağının bilgisini tutan ve arayüzün işleyişi gereği LeftSideButton adında tanımladığımız sınıftan üretilmiş nesneleri tutan tek boyutlu bir dizi değişkendir.
- cRow, cCol, gridNumber: Bu değişkenler integer (tam sayı) temel tipindedir. Kamera görüntüsünün ızgaralı yapısının oluşturulması için ilk değerlerini alırlar. Aldıkları bu ilk değerler, aracın açılış anında kamera görüntüsünün sadece tek çerçeveli olması için ayarlanmıştır.

3.3.2.3 stop( ) fonksiyonu

Bu fonksiyon, sadece yazılımın işleyişi sonlandırıldığında çalışacak olan ve normal iş akışı içerisinde asla çağırılmayacak olan standart bir processing fonksiyonudur. Bu yüzden, setup( ) ve draw( ) fonksiyonları çalışmaya başladıktan sonra, kullanıcı uygulamayı kapamadıkça aktif hale gelmeyen ve genel işleyişle hiçbir zaman ilişki

kurmayan bir fonksiyondur. Bunun için de, sistemin başlangıç ayarlarında tanımlanması gerekir. Yazılımın çalışması durdurulduğunda bu fonksiyon içerisinde iki işlev yerine getirilir:

Birincisi, AudioOutput sınıfı nesnelerini tutan otuz iki elemanlı outs dizisinin bütün elemanlarının sahip olduğu ses çıkış kanalı, AudioOutput sınıfının sahip olduğu close( ) fonksiyonu kullanılarak kapatılır.

İkinci olarak, setup( ) fonksiyonunda start( ) fonksiyonu ile çalıştırılan Minim sınıfının ve iş parçacıklarının, yazılımın herhangi bir hataya yol açılmadan kapatılması için, stop( ) fonksiyonu ile sonlandırılmasıdır.

3.3.2.4 setup( ) fonksiyonu

Yukarıda anlatılan tanımlamalardan sonra, ilk olarak setup( ) fonksiyonu çalışır. Bu fonksiyon, sadece bir kere uygulanmak üzere temel dört işleve sahiptir.

3.3.2.4.1 Kullanıcı arayüzünün boyutlarının belirlenmesi

Sistem çalışmaya başladıktan sonra hiç değişmeyecek bir şekilde kullanıcı arayüzünün boyutları belirlenir. Aracın arayüzünün boyutları yatayda 800 piksel, dikeyde 600 piksel olarak ayarlanmıştır. (Şekil 3.17)

Şekil 3.17 Kullanıcı arayüzünün boyutları

3.3.2.4.2 Kamera aygıtı ile ilgili ilk ayarlamaların yapılması

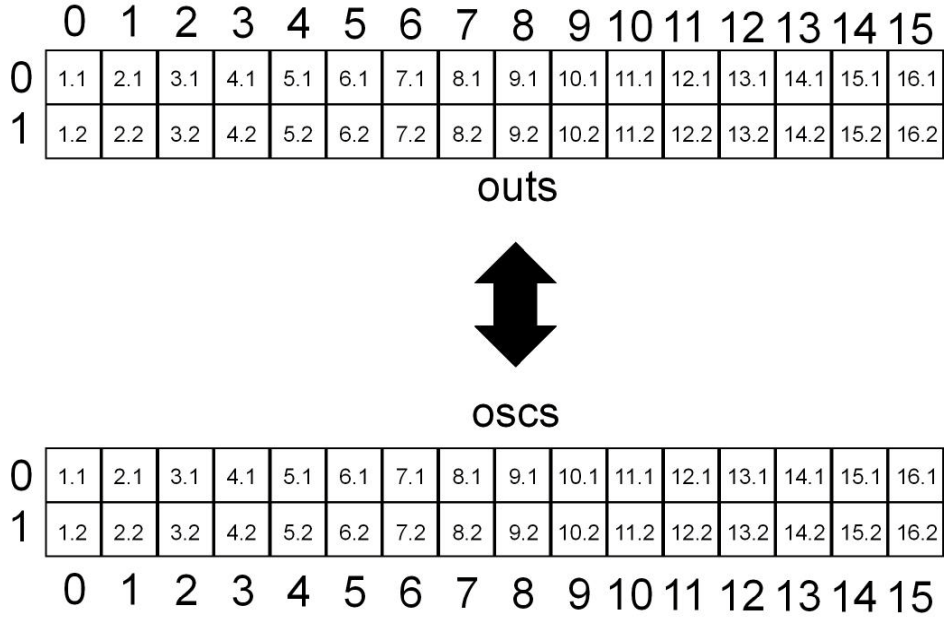
setup( ) fonksiyonunun bir sonraki işlevi, kamera aygıtı ile ilişki kurmamızı sağlayan ve global olarak tanımlanan Capture türündeki cam değişkenine bağlanmak üzere yeni bir Capture nesnesi üretilmesidir. Bunun ardından, bu değişkenin içeriğini oluşturan yakalanmış görüntülerin boyutları belirlenir. Yakalanan bu görüntülerin boyutları 256*256 piksel boyutlarında olmak üzere ayarlanır.

3.3.2.4.3 Ses sinyalleri ile ilgili ilk ayarlamaların yapılması

Ses sinyalleri ile ilgili olarak setup( ) fonksiyonu içerisindeki ilk ayarlama, ddf.minim paketinde bulunan Minim sınıfının start( ) yönteminin kullanılmasıdır. Bu fonksiyon ddf.minim paketi ile yapılacak olan bütün işlemler için gerekli olan hazırlığı sağlar.

İkinci olarak, AudioOutput sınıfı nesnelerini tutması için yaratılmış olan outs dizi değişkenine AudioOutput dizisi üretilerek bağlanır. Burada önemli olan, outs dizisinin boyutlarıdır. Görüntü kanalından, kullanıcı kararları ile on altı çerçeve görüntü alınabildiği ve her bir çerçeveden iki ses kanalı üretilebildiği düşünülerek, outs dizisinin yatay boyutunun on altı, dikey boyutunun ise iki olan bir matris olması gerektiğine karar verilmiştir.

Üçüncü olarak, outs dizi değişkeni için yapılan ayarlamaların, oscs değişkeni için de yapılması gerekmektedir. Oscillator sınıfı nesnelerini tutması için yaratılmış olan outs dizi değişkenine Oscillator dizisi üretilerek bağlanır. Kullanıcı kararlarına bağlı olarak otuz iki ses kanalının yaratılabileceği düşünülürse bu otuz iki değeri kendi içerisinde tutabilmesi için bu boyutta bir dizi değişkeni olması gerekir. Ancak oscs dizisinin, outs dizisi ile paralel çalışması gereken bir dizi olduğunu ve outs dizisinin yapısını göz önünde tutarsak, oscs dizisinin otuz iki elemanlı ve on altı basamak yatay, iki basamak dikey olarak düzenlenmiş bir matris olması gerektiği görülür. (Şekil 3.18)



Şekil 3.18 outs ve oscs dizilerinin paralel tanımlanmış matris yapıları

Dördüncü olarak, her bir outs dizi elemanının taşıdığı AudioOutput nesnesinin bir çıkış kanalına sahip olması gerekliliğidir. Bunun için, Minim sınıfının getLineOut( ) yöntemi her bir dizi elemanı için çağırılır ve bu elemanların içindeki AudioOutput nesnesine bu yöntem ile bir ses çıkış kanalı bağlanır. Burada önemli olan iki nokta vardır. Birincisi, ses çıkış kanallarının stereo özelliğe sahip olması ve buffersize özelliğinin de 512 olmasıdır. Stereo özellik kullanıcının her iki hoparlörü de istediği gibi kullanması konusunda ki kararımız gereği bulunacaktır. Buffersize özelliğinin de 512 olması düzgün çalışan bir ses çıkış kanalının kaliteli ses üretmesi için gerekli olacak en düşük değer olmasından dolayıdır.

Beşinci olarak, her bir oscs dizi elemanının taşıdığı Oscillator nesnesinin bir ses sinyal değerine sahip olması için gerekenlerin yapılmasıdır. Burada önemli olan nokta; kullanıcı, sistem çalışmaya başladıktan sonra görüntü çerçevelerinin sayısını değiştirerek ses kanallarının çıkış sayısını sürekli olarak değiştirebileceğidir. Bu yüzden otuz iki ses kanalı ilk başlangıçta üretilmiş ve bu kanallara başlangıç değeri olarak sinyallerin atanmış olması gerektiği düşünülmüştür. Ses kanallarının ve ses sinyali üretmenin sistem kaynaklarını, görüntü verilerini işleme ile karşılaştırıldığında oldukça az tüketmesi bu yolu seçmemiz de bir etken olmuştur. Buradan yola çıkarak outs dizisinde yapıldığı gibi, otuz iki oscs dizisi elemanına başlangıç sinyal değerleri

atanmıştır. Bu değerler; 0 frekans, 0 ses yüksekliği (amplitude) ve outs dizisi elemanları olan çıkış kanallarıyla uygun bir şekilde çalışabilmesi için 512 değerinde buffersize olarak ayarlanmıştır. Bunun yanında başlangıç olarak üretilen bu ses değerlerinin pulswave dalga türü ile üretilmesine karar verilmştir.

Altıncı olarak, oscs dizisinde bulunan ses dalgalarının, outs dizisine atanması durumudur. Bu da otuz iki outs elemanı için teker teker gerçekleştirilir. Bu aşamanın da sonuna gelindiğinde artık otuz iki adet çıkışa atanmış, otuz iki adet ses sinyali hazır olarak bulunur.

3.3.2.4.4 Değişkenlerini tanımladığımız sınıfların ve nesnelerinin yaratılması

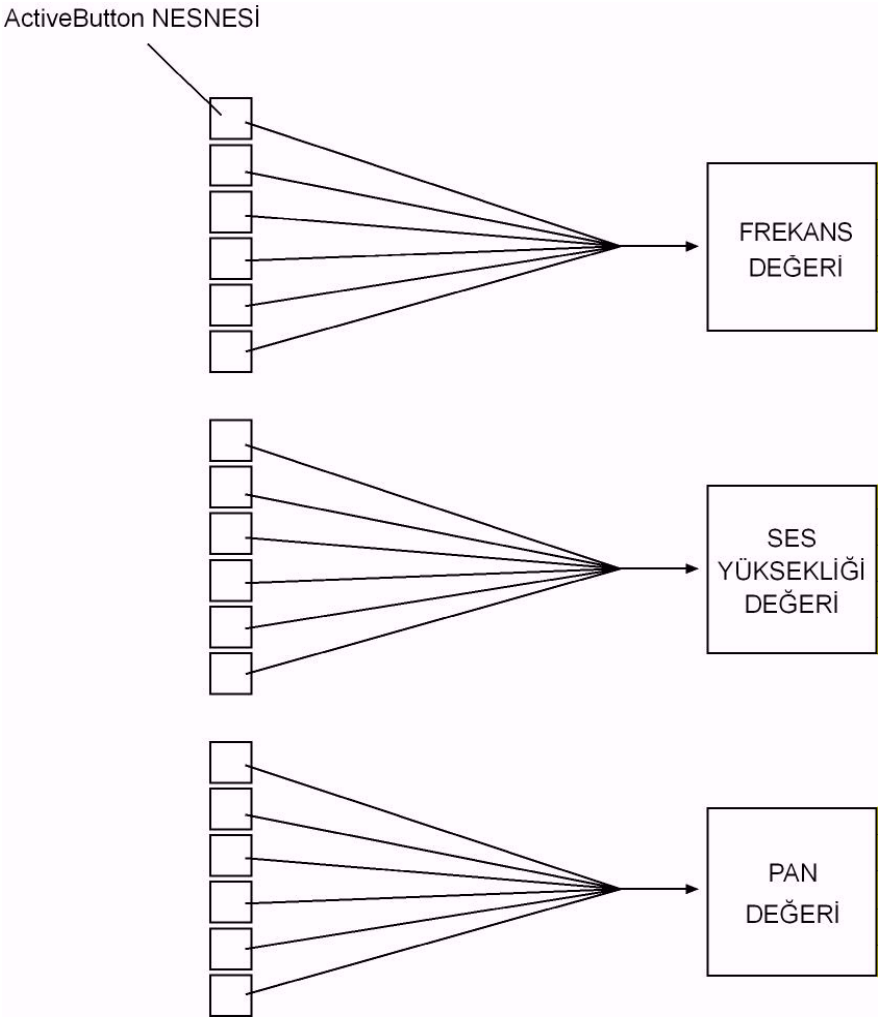
Çalışmamızda global değişkenlerin tanımlanmasının anlatıldığı kısımda, ilgili ses ve görüntü parametre değerlerinin tutulmasına yarayan bu değişkenlerin sınıf türünde değerleri tuttuğu belirtilmiştir. Değer tutacak olan bu değişkenlerin sınıf türünde tanımlanması, aracın sahip olduğu sistem ile ilgilidir. Araç kamera kaynaklı verileri ses sinyallerine dönüştürme görevini yerine getirirken, kullanıcı kontrolleri ile ilgili grafik arayüzün ve arka tarafta çalışan sistemin paralel ve düzenli çalışması gerekmektedir. Bu nedenle, değer taşıyan değişkenler herhangi bir temel tip olarak değil, sınıf yapısında yaratıldılar. Böylece, sınıfın özellikleri ile oluşturulan arayüz elemanlarıyla ve sınıf yöntemleri ile oluşturulan kullanıcı etkileşimleriyle paralel ve düzenli çalışmasını sağlamak amacıyla, ses ve görüntü parametre değerlerinin de sınıfın özellikleri olarak yaratılmasına karar verildi.

Ürettiğimiz yazılım, global değişkenlerini tanımladığımız sınıfların nesnelerini setup( ) fonksiyonunda tanımlar. Sırasıyla tanımladığımız sınıflar ve setup( ) fonksiyonu içerisinde yarattığımız nesneler şunlardır:

- ActiveButton: Bu sınıf, görüntü yakalayan kameradan alınan görüntülerin işlenmesi ile elde edilen ve kamera kaynaklı olarak yaratılacak olan ses sinyallerini belirlemede kullanılacak altı adet parametrenin (kırmızı, yeşil, mavi, renk, doygunluk, parlaklık) kullanıcı kontrolleri ile oranları belirlendikten sonraki değerlerini taşıyan integer temel tipinde bir özelliği barındırır. Bunun yanında, bu altı parametreden hangilerinin ses sinyaline aktarılacağının değerini taşıyan boolean temel tipinde bir özellik vardır. Bunların haricinde,

ActiveButton sınıfından üretilen nesnelerin kullanıcı arayüzü içerisinde tanımlı olduğu lokasyonları belirten integer temel tipinde özellikler ve bu butonun kullanıcı tarafından seçilip seçilmediğinin bilgisini renk ile geri dönecek olan color temel tipinde bir özellik vardır. Ek olarak, kullanıcı arayüzünün çalışması için gerekli olan yöntemleri ve bir adet yapımcı yöntemini barındırır. Bu yapımcı yönteminde, ses sinyallerine aktarılacak olan görüntü parametrelerinin başlangıç değeri olarak 0 değeri atanır.

Bu sınıftan setup( ) fonksiyonu içerisinde on sekiz tane nesne üretilir. Üç adet ses parametresini ilgilendiren, her biri için altı görüntü parametresi olmak üzere, toplam on sekiz tane ActiveButton nesnesi gereklidir. (Şekil 3.19)

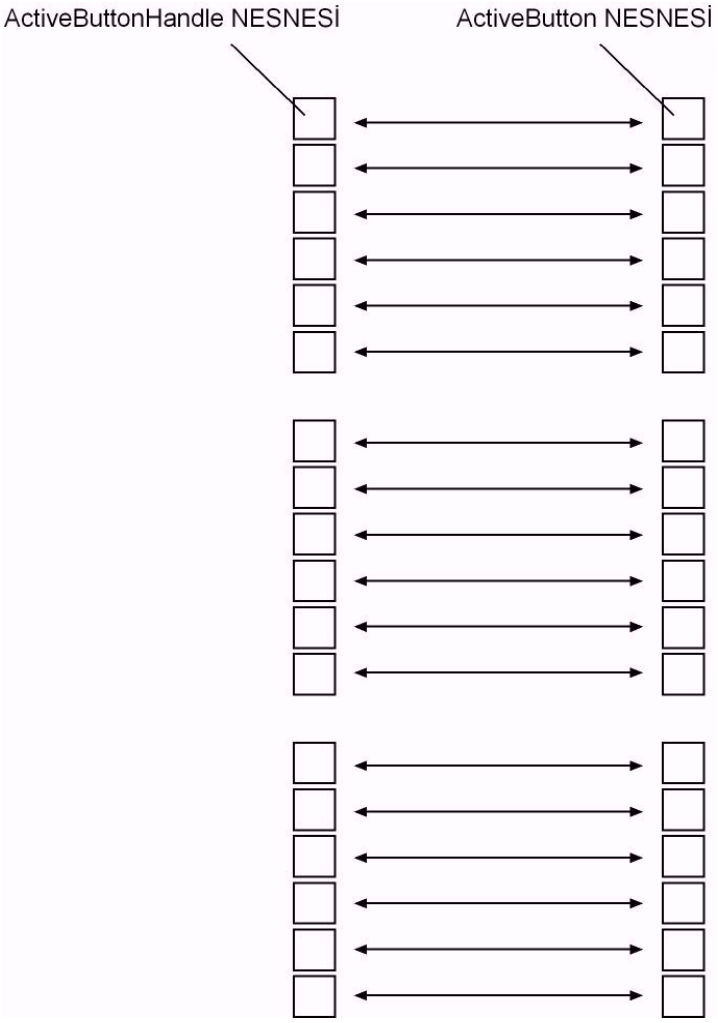


Şekil 3.19 Üç ses değerinin oluşması için ActiveButton sınıfından on sekiz tane nesne üretilmesinin gerekliliği

Bu ActiveButton sınıfından üretilen on sekiz nesne, global olarak tanımladığımız ve on sekiz eleman boyutuna sahip activeButtons dizi değişkeninde tutulur.

- ActiveButtonHandle: Bu sınıf, görüntülerin işlenmesi ile elde edilen ve kamera kaynaklı olarak yaratılacak olan ses sinyallerini belirlemede kullanılacak altı adet parametrenin (kırmızı, yeşil, mavi, renk, doygunluk, parlaklık) kullanıcı kontrolleri ile hangi oranda ses sinyal değerlerine katılacağının bilgisini taşıyan integer temel tipinde bir özelliği barındırır. Bunun yanında, oranların belirlenmesinde kullanıcının etkileşime gireceği butonların kullanıcı tarafından güncel olarak seçilip seçilmediğinin bilgisini taşıyan boolean temel tipinde bir özellik taşır. Bunların haricinde, ActiveButtonHandle sınıfından üretilen nesnelerin kullanıcı arayüzü içerisinde tanımlı olduğu lokasyonları belirten integer temel tipinde özellikler vardır. Ek olarak, kullanıcı arayüzünün çalışması için ek özellikleri ve gerekli olan yöntemleri ve bir adet yapımcı yöntemini barındırır. Bu yapımcı yönteminde, ses sinyallerine katılacak olan görüntü parametrelerinin başlangıç oranın değeri 0 olarak atanır.

  Bu sınıftan setup( ) fonksiyonu içerisinde on sekiz tane nesne üretilir. On sekiz adet ActiveButton nesnesinin oranlarını belirlemek için yine on sekiz adet ActiveButtonHandle nesnesi gereklidir. (Şekil 3.20)



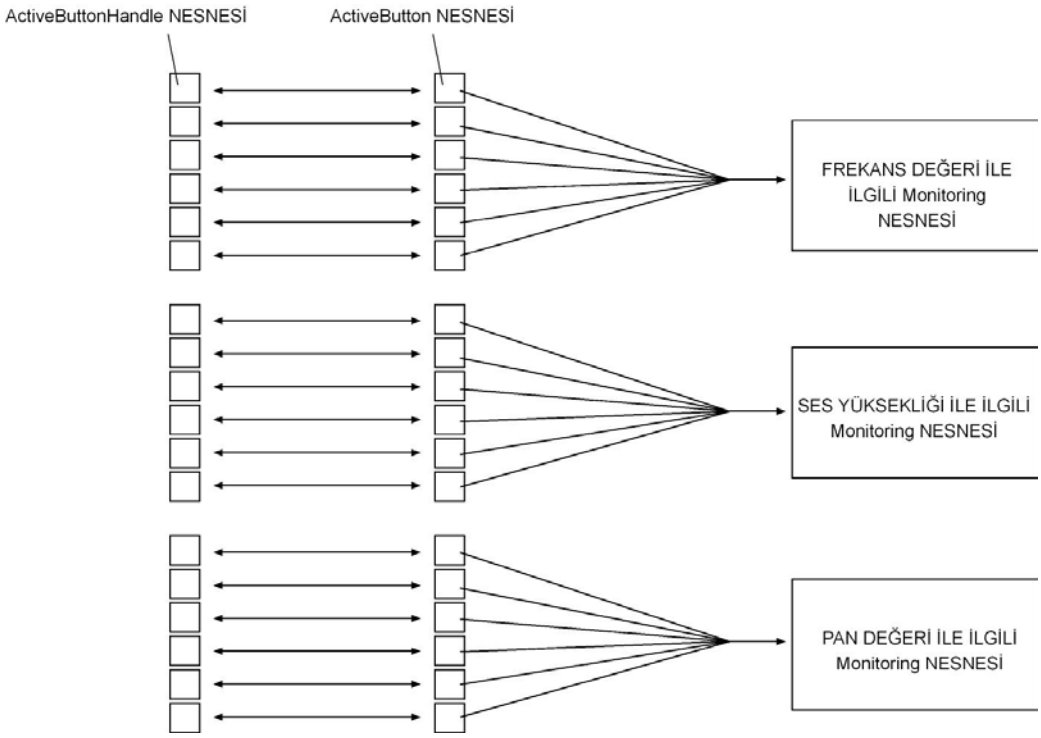
Şekil 3.20 On sekiz adet ActiveButton nesnesine karşılık olarak on sekiz adet ActiveButtonHandle nesnesinin gerekliliği

Bu ActiveButtonHandle sınıfından üretilen on sekiz nesne, global olarak tanımladığımız ve on sekiz eleman boyutuna sahip activeButtonHandles dizi değişkeninde tutulur.

- Monitoring: Bu sınıf, kamera kaynaklı görüntülerin işlenmesi ile elde edilen ses sinyallerini belirlemede kullanılacak altı adet parametrenin (kırmızı, yeşil, mavi, renk, doygunluk, parlaklık) kullanıcı kontrolleri ile değerlerinin belirlenmesinden sonra aldıkları son değerlerin bilgisini taşıyan integer temel tipinde özellikler barındırır. Bunun yanında, oranların belirlenmesinde kullanıcının etkileşime gireceği butonların kullanıcı tarafından güncel olarak seçilip seçilmediğinin bilgisini taşıyan boolean temel tipinde bir özellik taşır. Bunların haricinde, Monitoring sınıfından üretilen nesnelerin kullanıcı arayüzü

içerisinde tanımlı olduğu lokasyonları belirten integer temel tipinde özellikler vardır. Ek olarak, kullanıcı arayüzünün çalışması için ek özellikleri ve gerekli olan yöntemleri ve bir adet yapımcı yöntemini barındırır.

Bu Monitoring sınıfından setup( ) fonksiyonu içerisinde üç tane nesne üretilir. Üç adet Monitoring nesnesi üç ses sinyal değerini karşılarken, karşılıklı olarak on sekiz adet ActiveButton nesnesi ile ilişkiye girer. (Şekil 3.21)



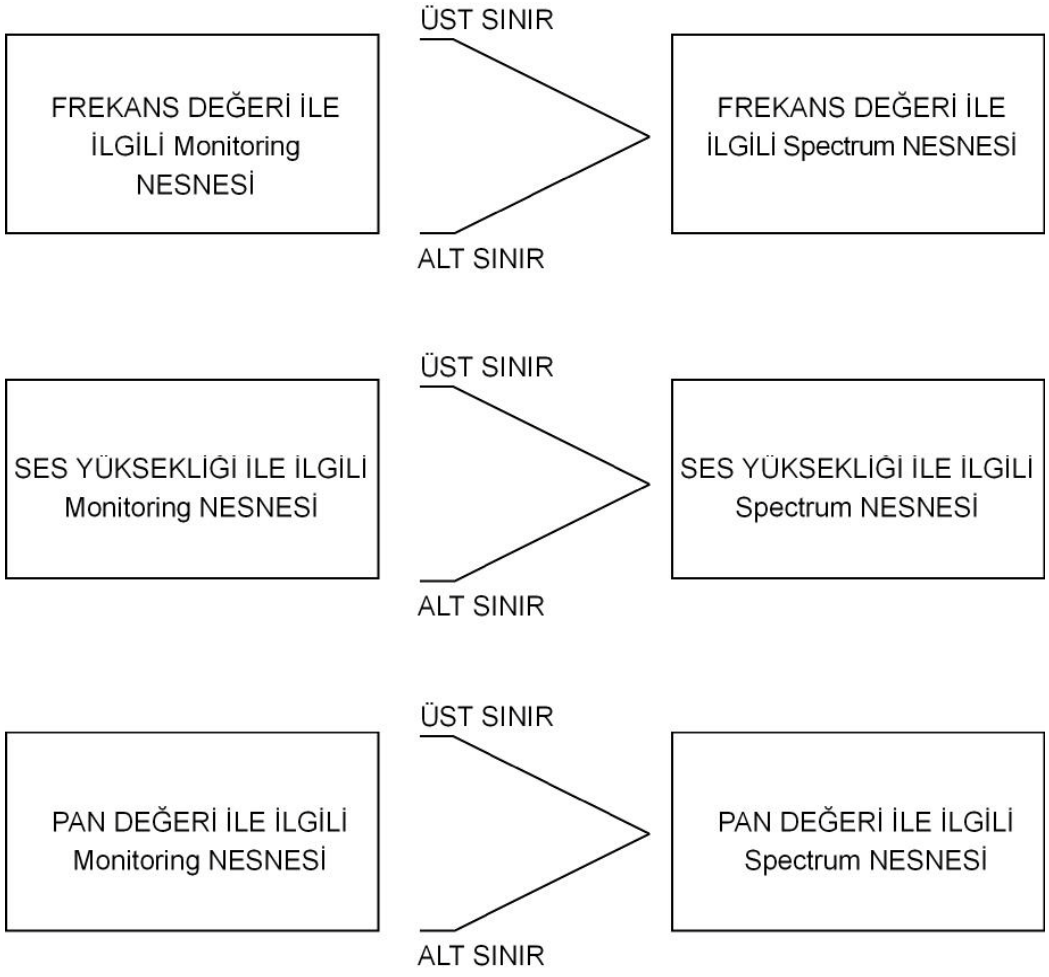
Şekil 3.21 Monitoring nesnelerinin karşılıklı ilişkide bulunduğu ActiveButton ve ActiveButtonHandle nesneleri

Bu Monitoring sınıfından üretilen üç tane nesne, global olarak tanımladığımız ve üç eleman boyutuna sahip monitorings dizi değişkeninde tutulur.

- Spectrum: Bu sınıf, Monitoring nesnelerinin tuttuğu değerlerin spektrum aralıklarını belirlemek için kullanılan ve kullanıcı kontrolleri ile alt ve üst sınırların değerlerinin bilgisini taşıyan integer temel tipinde özellikler barındırır. Bunun yanında, aralıkların belirlenmesinde kullanıcının etkileşime gireceği butonların kullanıcı tarafından güncel olarak seçilip seçilmediğinin bilgisini taşıyan boolean temel tipinde bir özellik taşır. Bunların haricinde, Spectrum

sınıfından üretilen nesnelerin kullanıcı arayüzü içerisinde tanımlı olduğu lokasyonları belirten integer temel tipinde özellikler vardır. Ek olarak, kullanıcı arayüzünün çalışması için ek özellikleri ve gerekli olan yöntemleri ve bir adet yapımcı yöntemini barındırır.

Bu Spectrum sınıfından setup( ) fonksiyonu içerisinde üç tane nesne üretilir. Üç adet Spectrum nesnesi, üç adet Monitoring nesnesinin taşıdığı değerlerin sınır aralıklarını belirler. Bunun için karşılıklı olarak üç adet Monitoring nesnesi ile ilişkiye girer. (Şekil 3.22)

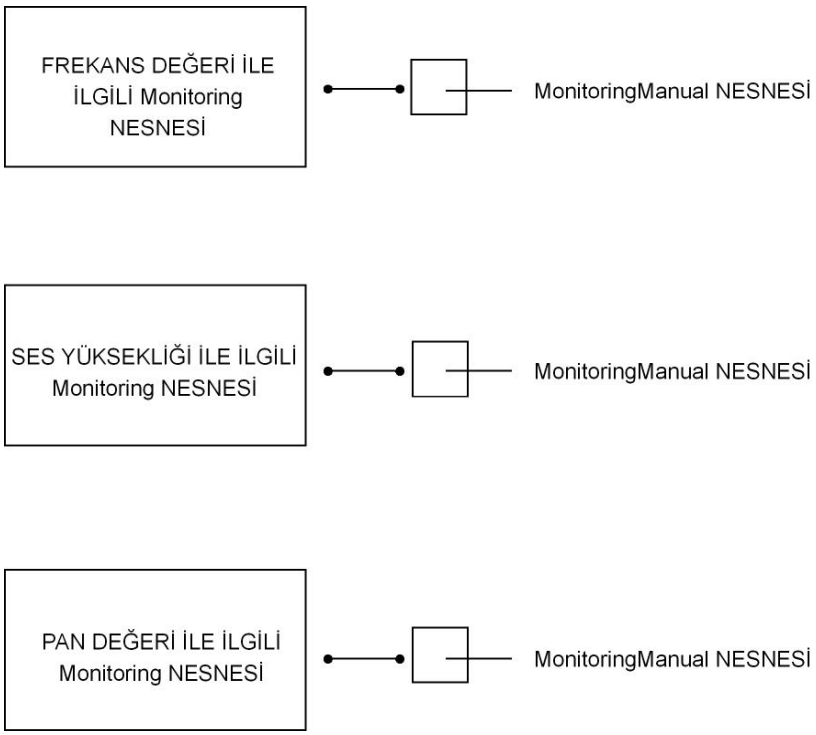


Şekil 3.22 Spectrum nesnelerinin karşılıklı ilişkide bulunduğu Monitoring nesneleri

Bu Spectrum sınıfından üretilen üç tane nesne, global olarak tanımladığımız ve üç eleman boyutuna sahip spectrums dizi değişkeninde tutulur.

- MonitoringManual: Bu sınıf, Monitoring nesnelerinin tuttuğu değerlerin kamera kaynaklı mı yoksa el ile mi girileceğinin bilgisini taşıyan boolean temel tipinde özellik barındırır. Bunun yanında, MonitoringManual sınıfından üretilen nesnelerin kullanıcı arayüzü içerisinde tanımlı olduğu lokasyonları belirten integer temel tipinde özellikler vardır. Ek olarak, kullanıcı arayüzünün çalışması için ek özellikleri ve gerekli olan yöntemleri ve bir adet yapımcı yöntemini barındırır.

Bu MonitoringManual sınıfından setup( ) fonksiyonu içerisinde üç tane nesne üretilir. Üç adet MonitoringManual nesnesi, üç adet Monitoring nesnesinin taşıdığı değerlerin kamera kaynaklı ya da el ile girileceğinin bilgisini taşıyacaktır. Bunun için üç adet Monitoring nesnesi ile ilişkiye girer. (Şekil 3.23)



Şekil 3.23 MonitoringManual nesnelerinin Monitoring nesneleri ile ilşkilileri

Bu MonitoringManual sınıfından üretilen üç tane nesne, global olarak tanımladığımız ve üç eleman boyutuna sahip monitoringManuals dizi değişkeninde tutulur.

- LeftSideButton: Bu sınıf, arayüz içerisinde birçok defa kullanılan ve farklı işlevlere sahip olan ancak görsel ve temel etkileşim anlamında kullanıcının standart bir kontrole sahip olduğu arayüz elemanları için tanımlandı. Kullanıcı arayüzü içerisinde tanımlı oldukları lokasyonları belirten integer temel tipinde özellikler vardır. Ek olarak, kullanıcı arayüzünün çalışması için ek özellikleri ve gerekli olan yöntemleri ve bir adet yapımcı yöntemini barındırır. Bu arayüz elemanlarını tutan, işlevlerine göre tanımlanmış farklı global değişkenler vardır. Bu global değişkenler şunlardır:
    - gridNumberSelections: Arayüzde, kamera kaynağının kaç çerçeve olacağının bilgisinin girilmesi için yaratılmış, üç adet LeftSideButton nesnesini tutan ve üç eleman boyutuna sahip dizi değişkenidir.
    - signalNumberSelections: Arayüzde, kamera kaynağının her bir çerçevesinin sahip olduğu iki ses kanalından hangisinin güncel seçim halinde olacağının bilgisinin girilmesi için yaratılmış, iki adet LeftSideButton nesnesini tutan ve iki eleman boyutuna sahip dizi değişkenidir.
    - signalKindSelections: Arayüzde, ses sinyallerinin hangi dalga türünde üretileceğinin bilgisinin girilmesi için yaratılmış, beş adet LeftSideButton nesnesini tutan ve beş eleman boyutuna sahip dizi değişkenidir.
    - previewbutton: Arayüzde, kamera ile yakalanan görüntünün kulanıcı tarafından da görülmesinin istenip istenmediğinin bilgisinin girilmesi için yaratılmış, bir adet LeftSideButton nesnesini tutan global değişkendir.
    - gridbutton: Arayüzde, kamera ile yakalanan görüntünün kulanıcı tarafından ızgaralara ayrılmış şekilde görülmesinin istenip istenmediğinin bilgisinin girilmesi için yaratılmış, bir adet LeftSideButton nesnesini tutan global değişkendir.

- linebutton: Arayüzde, kamera ile yakalanan görüntünün parametre değerlerini gösteren ActiveButton nesneleri ile ses sinyal değerlerini gösteren monitoring nesneleri arasında bulunan ilişkinin dinamik olarak çizilen çizgilerle görsel olarak da görülmesinin istenip istenmediğinin bilgisinin girilmesi için yaratılmış, bir adet LeftSideButton nesnesini tutan global değişkendir.
- portabutton: Arayüzde, üretilen ses sinyallerinin portamento değerlerinin olup olmayacağının bilgisinin girilmesi için yaratılmış, bir adet LeftSideButton nesnesini tutan global değişkendir.

- PortamentoTool: Bu sınıf, üretilen ses sinyallerinin portamento değerlerinin kullanıcı kontrolü ile belirlenen bilgisini taşıyan integer temel tipinde özellikler barındırır. Bunun yanında, güncel olarak seçilip seçilmediğinin bilgisini taşıyan boolean temel tipinde bir özellik taşır. Bunların haricinde, PortamentoTool sınıfından üretilen nesnelerin kullanıcı arayüzü içerisinde tanımlı olduğu lokasyonları belirten integer temel tipinde özellikler vardır. Ek olarak, kullanıcı arayüzünün çalışması için ek özellikleri ve gerekli olan yöntemleri ve bir adet yapımcı yöntemini barındırır.

  Bu PortamentoTool sınıfından setup( ) fonksiyonu içerisinde bir tane nesne üretilir ve bu PortamentoTool sınıfından üretilen bir adet nesne, global olarak tanımladığımız portamento değişkeninde tutulur.
- Synth: Yukarıda belirtilen sınıflar, arayüzde sürekli bulunan ve sadece değerleri değişen nesnelerin yaratıldığı sınıflardır. Bu sınıflar, kullanıcı konrolü ile otuz iki kanala çıkabilen ses dalga özellikleri ile ilgili olan her şeyi barındırmaktadırlar. Ancak kendi sınıflarında sadece tek bir ses kanalı için barındırılan bu bilgiler, otuz iki ses kanalı için ayrı ayrı tutulabilmelidir.

  Bu problemin çözümü için bir veri yapısı düşünüldü. Bu veri yapısı, bütün kanallar için gerekli olan verileri üzerinde taşıyabilecek bir yeteneğe sahip olmalıdır. Bunun için toplam otuz iki elemana sahip ve her bir elemanda bir ses kanalı için ve ilgili ses kanalının bütün arayüz verileri için gerekli olan bütün bilgilerin bulunduğu bir dizi yaratılmasına karar verildi. Bunun için önce tek bir

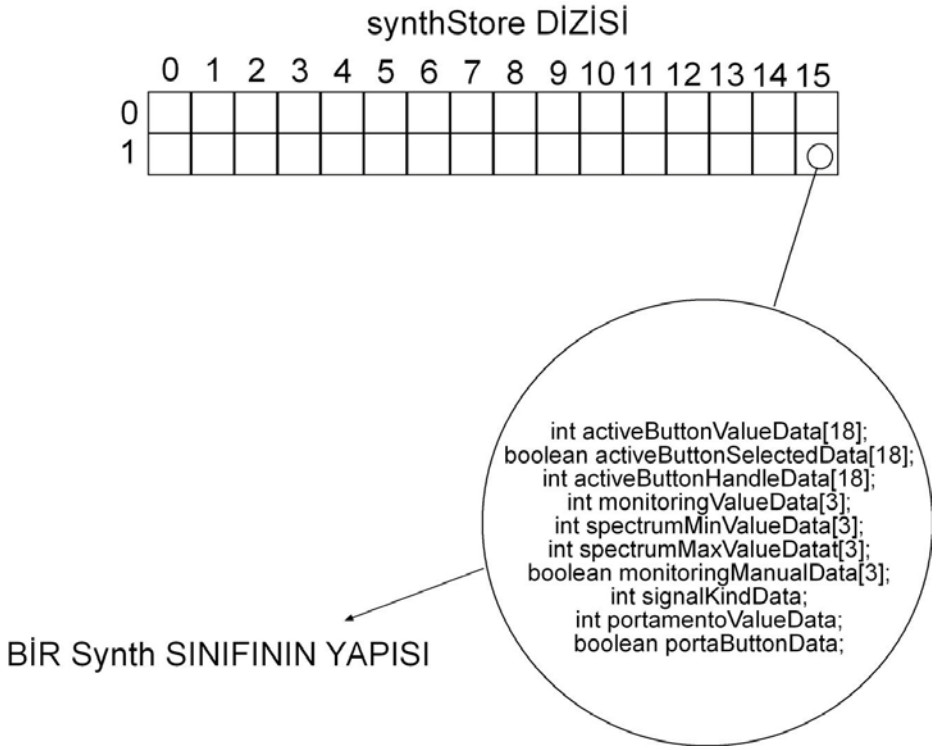
ses kanalının bütün bilgilerini taşıyan Synth isimli bir sınıf tanımlandı. Bu sınıfta sırasıyla şu özellikler vardır:

- on sekiz eleman boyutunda ve ActiveButton nesnelerinin değerini tutan activeButtonValueData isimli integer temel tipinde bir dizi değişkeni
- on sekiz eleman boyutunda ve ActiveButton nesnelerinin seçili olup olmadığının bilgisini tutan activeButtonSelectedData isimli boolean temel tipinde bir dizi değişkeni
- on sekiz eleman boyutunda ve ActiveButtonHandle nesnelerinin değerini tutan activeButtonHandleData isimli integer temel tipinde bir dizi değişkeni
- üç eleman boyutunda ve Monitoring nesnelerinin değerini tutan monitoringValueData isimli integer temel tipinde bir dizi değişkeni
- üç eleman boyutunda ve Spectrum nesnelerinin minimum değerini tutan spectrumMinValueData isimli integer temel tipinde bir dizi değişkeni
- üç eleman boyutunda ve Spectrum nesnelerinin maksimum değerini tutan spectrumMaxValueData isimli integer temel tipinde bir dizi değişkeni
- üç eleman boyutunda ve MonitoringManual nesnelerinin değerini tutan monitoringManualData isimli boolean temel tipinde bir dizi değişkeni
- signalKindSelections dizisinin hangi elemanının güncel seçim halinde olduğunun bilgisini taşıyan signalKindData isimli integer temel tipinde bir değişken
- PortamentoTool nesnesinin değerinin bilgisini taşıyan portamentoValueData isimli integer temel tipinde bir değişken
- portabutton değişkeninin değerinin bilgisini taşıyan portaButtonData isimli boolean temel tipinde bir değişken

Bu şekilde, Synth sınıfından türetilmiş her nesnenin, bir ses sinyal kanalı ile ilgili bütün parametre ve arayüz değerlerinin bilgisini taşıması sağlandı. Toplam otuz iki ses sinyal kanalı olduğuna göre, bu Synth sınıfından toplam otuz iki adet nesne üretilecektir. Bu nesnelerin saklanacağı yapı olarak, yine otuz iki eleman boyutunda ve Synth sınıfı türünde bir global değişken tanımlanmıştır.

synthStore isimli bu global değişken, outs ve oscs dizileri ile uyum halinde çalışacağından dolayı, onlar gibi yatayda on altı ve dikeyde iki elemanlı olmak üzere tasarlandı.

Bu yapı içinde synthStore isimli dizinin önemli bir görevi olmaktadır. Toplam otuz iki adet ses kanalı ile ilgili bütün parametre ve arayüz değerleri bu dizide tutulacaktır. Kullanıcı ses kanalları arasında herhangi bir geçiş yaptığında, yeni geçtiği ses kanalının parametrelerine göre arayüz hemen güncellenecek ve arayüz yeni seçilen ses kanalının bilgilerini gösterecektir. Yine kullanıcı, herhangi bir ses kanalının herhangi bir özelliğini değiştirdiğinde, bu değişiklik hemen synthStore dizisine kaydedilecektir. Bu şekilde düzenlenmiş bir yapıda, Synth sınıfı türünde elemanlara sahip olan synthStore dizisi oldukça merkezi bir öneme sahiptir. (Şekil 3.24)



Şekil 3.24 Synth sınıfı türündeki synthStore dizisinin yapısı ve Synth sınıfının özellikleri

Böylece global olarak tanımladığımız değişkenlerin tuttukları içeriklerin, sınıfların ve bu sınıfların nesnelerinin yaratılması ve bu nesnelere ilk değerlerinin atanması işlemi

sonlanmış olur. Yazılımın setup( ) fonksiyonunun sonuna gelindiğinde, aracın çalışması ve arayüzün kendisini oluşturarak kullanıcı etkileşimlerine imkan vermesi için bütün hazırlıklar sonlanmış olur. Yazılımın bir sonraki adımı; grafik arayüzün oluşturulmasını, kameradan alınan görüntülerin işlenmesini, işlenen bu verilerden ses sinyal parametrelerinin yaratılmasını ve bunlarla ses sinyallerinin oluşturulmasını sağlayan ve kullanıcının sonlandırmasına kadar döngü halinde devam edecek olan draw( ) fonksiyonun çalışmaya başlamasıdır.

3.3.3 Kullanıcı Grafik Arayüzünün Yaratılması

Yazılımın; çalışmaya başlaması, ilgili paketleri yüklemesi ve setup( ) fonksiyonunu sonlandırmasının ardından draw( ) fonksiyonu devreye girer. Bu fonksiyon, sistemin bundan sonraki bütün işleyişinin iskeletini oluşturmaktadır. draw( ) fonksiyonu dahilinde ilk görev, cam isimli Capture değişkeninde görüntü yakalayıcı aygıt olan kameradan alınmış hazır bir görüntüye sahip olup olmadığının sınanmasıdır. Eğer hazır bir görüntü yok ise, görüntü sağlanana kadar beklenir. İşlenmeye hazır bir görüntü olduğunda ise, GraphicUserInterface( ), ComputePixelValues( ) ve SoundGenerator( ) fonksiyonlarının sırayla çalıştırılmasına başlanır. (Şekil 3.16)

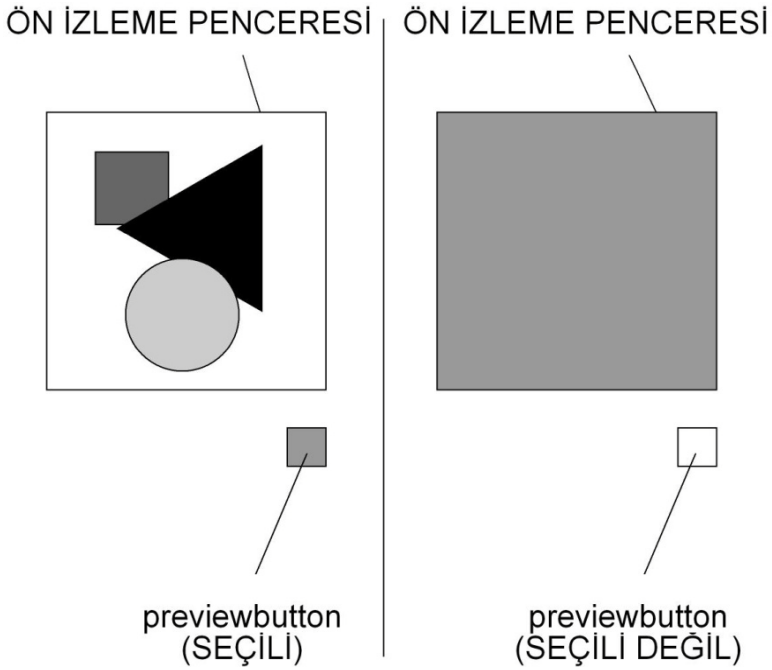
Kullanıcı grafik arayüzünün oluşturulması görevini üstlenen GraphicUserInterface( ) fonksiyonu, özellikle sistemin ürettiği güncel değerlerin kullanıcıya sunulması için gerekli olan arayüzün ve her türlü görsel öğenin yaratılması ve hazırlanması görevlerini yerine getirmekle yükümlüdür. Bu fonksiyon, draw( ) fonksiyonu içerisinde her döngünün başında ilk olarak çalıştırılır, çünkü ekranın her yeni adımda yeniden taranması gerekmektedir. Bu görev için birçok fonksiyonun çalıştırılması gereklidir.

Burada belirtilmesi gereken önemli bir nokta, kullanıcı arayüz elemanlarının oluşturulması sırasında, dışarıdan hiçbir imaj alınmaması ve sadece processing ortamının sağladığı standart fonksiyon ve özelliklerinin kullanılmasıdır. Böylece, hem sistem kaynaklarına ek yükler getirilmemiş, hem de aracın minimal çerçevesine uygun olan teknik bir yöntem seçilmiştir.

İlk olarak, arka plan piksellerinin rengi belirlenir. Böylece ekranın her taranmasında, ilk olarak arka plan pikselleri boyanır ve geri kalan bütün elemanlar bu arka plan rengi üzerine yerleştirilir. Processing ortamında, arka plan rengi için parametrik bir değer belirlenmesi gerekir. Aracın arka planının 0 (siyah) ile 255 (beyaz) değerleri arasında 120 değerinde bir gri renk olmasına karar verildi.

Ardından, çizgiler için renk ve kalınlık değerleri girildi. Dolgu renkleri olarak da, 200 değerinde bir gri renk seçildi. Bu değerler, tamamen başlangıç değerleri olup, daha sonra farklı arayüz öğeleri için yeniden tanımlanabilirler. Bu çizgi ve renk değerleri ile, kameradan alınan görüntünün ön izlemesinin (preview) yapılacağı alan çerçevelenerek belirlenir ve içi dolgu rengi ile doldurulur.

Kameradan gelen veriyi ön izleme ile seyretmek, kullanıcının kontrolüne verilmiştir. Eğer previewbutton değişkeninin selected özelliği ‘true’ durumda ise, yani previewbutton seçili ise, cam değişkeninde taşınan görüntü belirlenen bu alana yerleştirilir ve kullanıcı kameradan gelen görüntüyü seyredebilir. Eğer previewbutton değişkeninin selected özelliği 'false’ durumda ise, yani previewbutton seçili durumda değilse, cam değişkeninde belirlenen görüntü bu alana yerleştirilmez ve bu alan sadece belirlenen dolgu renginde gösterilir. Bu durum, kullanıcı için etkileşim çeşitliliği sağladığı gibi, aracın diğer işlevlerinin daha performanslı çalışabilmesi için sistem kaynaklarının daha az tüketilmesi için de bir seçenek olmuş oluyor. (Şekil 3.25)



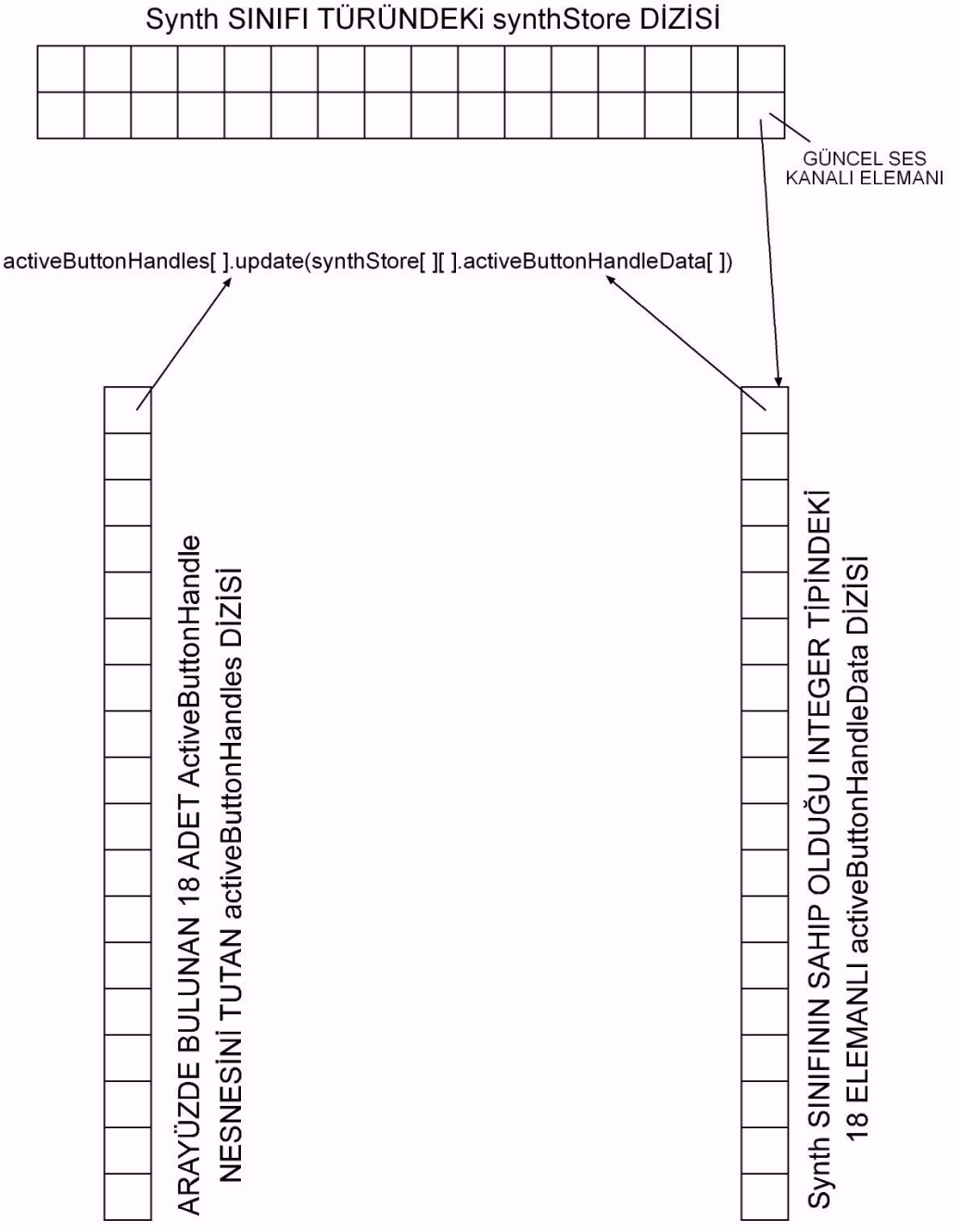
Şekil 3.25 previewbutton nesnesinin selected özelliğinin ‘true’ ya da ‘false’ olmasına göre ön izleme ekranının görünüşü

Bir diğer görev; arayüz nesnelerinin update( ) yöntemlerinin çalıştırılarak, aracın kullanıcı grafik arayüzünün aktif verilere göre güncellenmesini sağlamaktır. Bunun için arayüzü oluşturan sınıflardan yaratılan nesnelerin sahip olduğu update( ) fonksiyonları sırasıyla çalıştırılır:

- activeButtons[ ].update( ): Kullanıcı arayüzünde bulunan on sekiz adet ActiveButtons nesnesini taşıyan activeButtons dizisinin bütün elemanlarının update( ) yöntemi sırasıyla çalıştırılır. Her bir nesnenin sahip olduğu update( ) yöntemi, öncelikle nesnenin selected özelliğinin ‘true’ ya da ‘false’ olup olmadığını sınar. Bu özellik ‘true’ ise, yapımcı yöntem içerisinde tanımlanan ve seçili olma halini vurgulayan kırmızı-220, yeşil-30, mavi-60 değerlerinin oluşturduğu renk dolgu rengi olarak belirlenir. Eğer, selected özelliği false ise, gri-200 değerini taşıyan ve seçili olmama halini vurgulayan renk dolgu rengi olarak belirlenir. Dolgu renginin belirlenmesinden sonra çizgi kalınlığının değeri belirlenir. Bu değer, strokeWeight( ) fonksiyonun parametre değerine 2 girilerek yapılır. Ardından, belirlenmiş dolgu ve çizgi kalınlığı ile nesnenin yaratılması sırasında kullanılan yapımcı yöntemin lokasyon belirleyen özelliklere atadığı yatay ve dikey konum değerleri ile kenar uzunlukları

bilgisinin parametrelerini oluşturduğu, processing ortamının standart rect( ) yöntemi çağrılır.

- activeButtonHandles[ ].update( ): Kullanıcı arayüzünde bulunan on sekiz adet ActiveButtonHandle nesnesini taşıyan activeButtonHandles dizisinin bütün elemanlarının update( ) yöntemi sırasıyla çalıştırılır. activeButtonHandles dizisinin elemanlarının update( ) fonksiyonu, synthStore dizisinin güncel elemanının sahip olduğu, yani ızgaralı ekranda seçili olan görüntü kanalının yatay ve dikey sıra bilgisini taşıyan değerlerin işaret ettiği synthStore dizisi elemanının sahip olduğu, activeButtonHandleData[ ] dizi değişkeninin ilgili elemanının değerini giriş parametresi olarak alır. Burada önemli olan nokta, otuz iki kanal ses dalgasının bütün parametre niceliklerini taşıyan Synth sınıfı türündeki synthStore dizisinin hangi elemanının, yani hangi ses kanalının, ilgili anda arayüzde gösterileceğinin bilinmesidir. Güncel ses kanalı için, ses kanalının bütün bilgileri arayüz nesnelerine atanır. (Şekil 3.26)

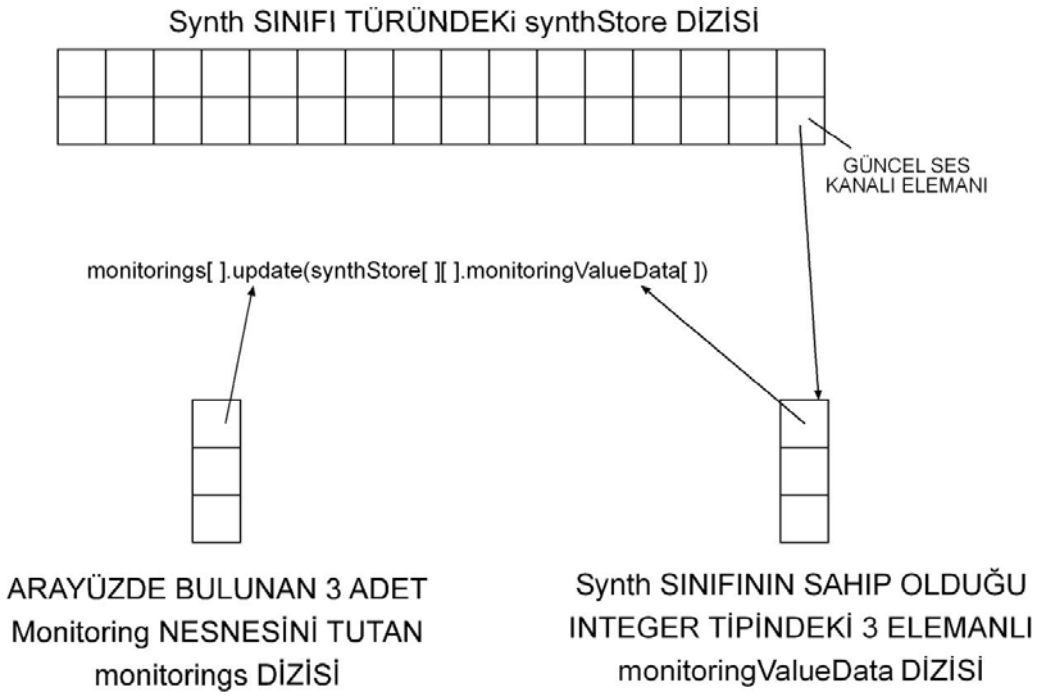


Şekil 3.26 activeButtonHandles dizisinin on sekiz elemanının update( ) yönteminin, güncel ses kanalı verilerinin giriş parametresi yapılması ile çalıştırılması

activeButtonHandles dizisi elemanlarının sahip olduğu update( ) fonksiyonunda ilk olarak, kullanıcıya ActiveButtons nesnelerini kullanarak kamera değerlerinin oranlarını belirlerken yardımcı olacak olan kılavuz çizgilerinin renk ve kalınlık bilgilerinin değerleri belirlenir. Bu değer, stroke( ) ve strokeWeight( ) standart fonksiyonları kullanılarak yapılır. stroke( ) fonksiyonunun parametre değerine 0

girilerek çizgi rengi olarak siyah belirlenir. strokeWeight( ) fonksiyonuna da, ActiveButtons nesnelerinde olduğu gibi 2 değeri girilir. Ardından line( ) standart fonksiyonu ile kılavuz çizgileri çizilir. Kulpun yatay konumunun güncel değere göre ne olacağını belirlemek için update( ) fonksiyonuna giriş parametresi olarak alınan değer kullanılır. Bu değerin kullanılmasıyla, computeHandleValueX( ) adlı sınıf yöntemi çağırılarak kulpun yatay konum değeri hesaplanır. Çizgi rengi gri-80 değerinde, dolgu rengi gri-200 değerinde belirlenir ve standart rect( ) fonksiyonu ile kulp arayüze çizilir. ActiveButtonHandle nesnesinden yaratılan kulp ile bunun değer olarak kaynaklandığı ActiveButton nesnesi arasında kullanıcıya ilişkiyi daha rahat görebilmesini sağlayacak yeni bir çizgi daha çizilir.

- monitorings[ ].update( ): Kullanıcı arayüzünde bulunan üç adet Monitoring nesnesini taşıyan monitorings dizisinin bütün elemanlarının update( ) yöntemi sırasıyla çalıştırılır. monitorings dizisinin elemanlarının update( ) fonksiyonu, synthStore dizisinin güncel elemanının sahip olduğu, yani ızgaralı ekranda seçili olan görüntü kanalının yatay ve dikey sıra bilgisini taşıyan değerlerin işaret ettiği synthStore dizisi elemanının sahip olduğu, monitoringValueData dizi değişkeninin ilgili elemanının değerini giriş parametresi olarak alır. Burada önemli olan nokta, otuz iki kanal ses dalgasının bütün parametre niceliklerini taşıyan Synth sınıfı türündeki synthStore dizisinin hangi elemanının, yani hangi ses kanalının, ilgili anda arayüzde gösterileceğinin bilinmesidir. Güncel ses kanalı için, ses kanalının bütün bilgileri arayüz nesnelerine atanır. (Şekil 3.27)

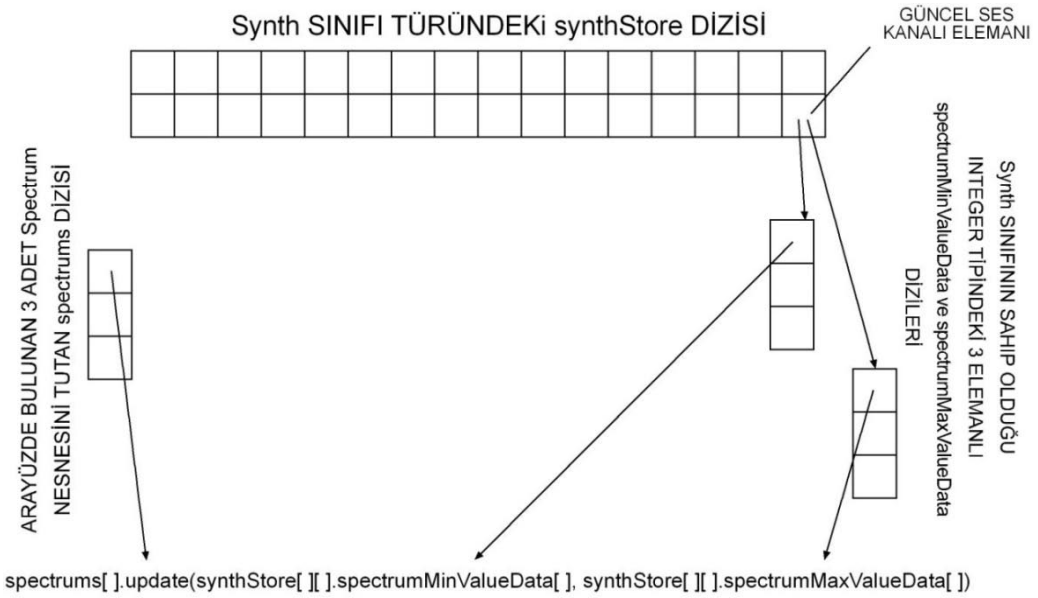


Şekil 3.27 monitorings dizisinin üç elemanının update( ) yönteminin, güncel ses kanalı verilerinin giriş parametresi yapılması ile çalıştırılması

monitorings dizisi elemanlarının sahip olduğu update( ) fonksiyonunda ilk olarak, kullanıcıya Monitoring nesnelerini kullanarak ses kanallarının sahip olduğu ve kamera kaynaklı olarak belirlenen üç özelliğin değerlerini görebilmeleri için yardımcı olacak olan kılavuz çizgilerinin renk ve kalınlık bilgilerinin değerleri belirlenir. Bu değer, stroke( ) ve strokeWeight( ) standart fonksiyonları kullanılarak yapılır. stroke( ) fonksiyonunun parametre değerine 0 girilerek çizgi rengi olarak siyah belirlenir. strokeWeight( ) fonksiyonuna 1 değeri girilir. Ardından line( ) standart fonksiyonu ile kılavuz çizgileri çizilir. Kulpun dikey konumunun, güncel değere göre ne olacağını belirlemek için update( ) fonksiyonuna giriş parametresi olarak alınan değer kullanılır. Bu değerin kullanılmasıyla, computeSoundValueY( ) adlı sınıf yöntemi çağırılarak kulpun dikey konum değeri hesaplanır. Çizgi rengi gri-80 değerinde, dolgu rengi gri-200 değerinde belirlenir ve standart line( ) ve rect( ) fonksiyonları ile kulp arayüze çizilir.

- spectrums[ ].update( ): Kullanıcı arayüzünde bulunan üç adet Spectrum nesnesini taşıyan spectrums dizisinin bütün elemanlarının update( ) yöntemi sırasıyla çalıştırılır. spectrums dizisinin elemanlarının update( ) fonksiyonu,

synthStore dizisinin güncel elemanının sahip olduğu, yani ızgaralı ekranda seçili olan görüntü kanalının yatay ve dikey sıra bilgisini taşıyan değerlerin işaret ettiği synthStore dizisi elemanının sahip olduğu, spectrumMinValueData ve spectrumMaxValueData dizi değişkenlerinin ilgili elemanlarının değerlerini giriş parametreleri olarak alır. Daha önce anlatılan monitorings dizisinin update( ) fonksiyonu tek giriş parametresi alırken, spectrums dizisinin update( ) fonksiyonunun iki giriş parametresi almasının nedeni; arayüzde bulunan spektum nesnelerinin işlev olarak hem üst hem de alt değer sınırlarını belirliyor olmalarıdır. Bu yüzden synthStore dizisine bağlı olan elemanın spectrumMinValueData ve spectrumMaxValueData olmak üzere iki dizisinin ilgili değerleri parametre olarak kullanılır. Burada önemli olan nokta, otuz iki kanal ses dalgasının bütün parametre niceliklerini taşıyan Synth sınıfı türündeki synthStore dizisinin hangi elemanının, yani hangi ses kanalının, ilgili anda arayüzde gösterileceğinin bilinmesidir. Güncel ses kanalı için, ses kanalının bütün bilgileri arayüz nesnelerine atanır. (Şekil 3.28)



Şekil 3.28 spectrums dizisinin üç elemanının update( ) yönteminin, güncel ses kanalı verilerinin giriş parametreleri yapılması ile çalıştırılması

spectrums dizisi elemanlarının sahip olduğu update( ) fonksiyonunda ilk olarak, kullanıcıya Spectrum nesnelerini kullanarak ses kanallarının sahip olduğu ve

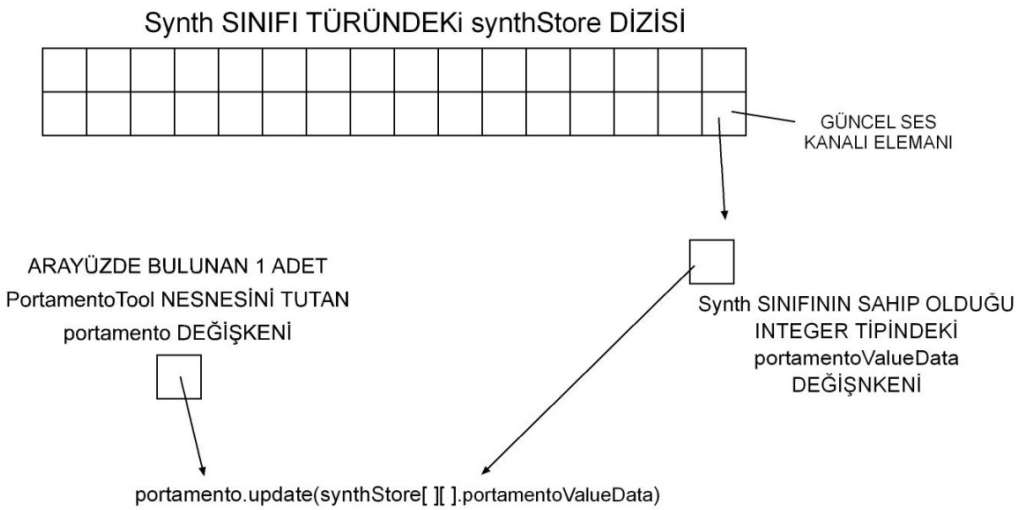
kamera kaynaklı olarak belirlenen üç özelliğin değerlerinin alt ve üst sınırlarını görebilmeleri için yardımcı olacak olan kılavuz çizgilerinin renk ve kalınlık bilgilerinin değerleri belirlenir. Bu değer, stroke( ) ve strokeWeight( ) standart fonksiyonları kullanılarak yapılır. stroke( ) fonksiyonunun parametre değerine 0 girilerek çizgi rengi olarak siyah belirlenir. strokeWeight( ) fonksiyonuna 1 değeri girilir. Ardından line( ) standart fonksiyonu ile kılavuz çizgileri çizilir. Kulpların dikey konumunun, güncel değere göre ne olacağını belirlemek için update( ) fonksiyonuna giriş parametresi olarak alınan değerler kullanılır. Bu değerler ile computeMinSpectrumY( ) ve computeMaxSpectrumY( ) adlı sınıf yöntemleri çağırılarak kulpların dikey konum değeri hesaplanır. Kulpların dikey konum bilgilerinin belirlenmesi ile standart line( ) ve rect( ) fonksiyonları kullanılarak kulplar arayüze çizilir.

- monitoringManuals[ ].update( ): Kullanıcı arayüzünde bulunan üç adet MonitoringManual nesnesini taşıyan monitoringManuals dizisinin bütün elemanlarının update( ) yöntemi sırasıyla çalıştırılır. Her bir nesnenin sahip olduğu update( ) yöntemi, öncelikle nesnenin value özelliğinin ‘true’ ya da ‘false’ olup olmadığını sınar. Bu özellik ‘true’ ise, yapımcı yöntem içerisinde tanımlanan ve seçili olma halini vurgulayan kırmızı-220, yeşil-30, mavi-60 değerlerinin oluşturduğu renk dolgu rengi olarak belirlenir. Eğer, selected özelliği false ise, gri-200 değerini taşıyan ve seçili olmama halini vurgulayan renk dolgu rengi olarak belirlenir. Dolgu renginin belirlenmesinden sonra çizgi kalınlığının değeri belirlenir. Bu değer, strokeWeight( ) fonksiyonun parametre değerine 2 girilerek yapılır. Ardından, belirlenmiş dolgu ve çizgi kalınlığı ile nesnenin yaratılması sırasında kullanılan yapımcı yöntemin lokasyon belirleyen özelliklere atadığı yatay ve dikey konum değerleri ile kenar uzunlukları bilgisinin parametrelerini oluşturduğu, processing ortamının standart rect( ) yöntemi çağrılır.
- signalKindSelections[ ].update( ): signalKindSelections dizisi sinyal dalga türünü belirlemek üzere kullanılan beş adet LeftSideButton nesnesinden oluşturulmuştur. Bu dizinin update( ) fonksiyonları çalıştırılmadan önce, synthStore dizisinin güncel olan kanal bilgisini taşıyan yatay ve dikey parametre değerlerinin işaret ettiği dizi elemanının sahip olduğu signalKindData

değişkeninin sınaması yapılarak, hangi signalKindSelections elemanının seçili olduğu bulunur. Bu belirlemeden sonra sırasıyla bütün signalKindSelections dizisi elemanlarının update( ) fonksiyonu çalıştırılır. signalKindSelections dizisi LeftSideButton nesnelerini tuttuğu için, bu update( ) fonksiyonu LeftSideButton sınıfına ait olan bir fonksiyondur. Bu fonksiyon içerisinde öncelikle, nesnenin selected özelliğinin 'true' ya da 'false' olup olmadığını sınar. Bu özellik 'true' ise, yapımcı yöntem içerisinde tanımlanan ve seçili olma halini vurgulayan kırmızı-220, yeşil-30, mavi-60 değerlerinin oluşturduğu renk dolgu rengi olarak belirlenir. Eğer, selected özelliği false ise, gri-200 değerini taşıyan ve seçili olmama halini vurgulayan renk dolgu rengi olarak belirlenir. Dolgu renginin belirlenmesinden sonra çizgi kalınlığının değeri belirlenir. Bu değer, strokeWeight( ) fonksiyonun parametre değerine 2 girilerek yapılır. Ardından, belirlenmiş dolgu ve çizgi kalınlığı ile nesnenin yaratılması sırasında kullanılan yapımcı yöntemin lokasyon belirleyen özelliklere atadığı yatay ve dikey konum değerleri ile kenar uzunlukları bilgisinin parametrelerini oluşturduğu, processing ortamının standart rect( ) yöntemi çağrılır.

- portabutton.update( ): Bu değişken, ses dalgalarının portamento özelliklerinin açık olup olmamasının bilgisini taşıyan arayüz nesnesi olmak üzere LeftSideButton sınıfından üretilmiştir. Bu değişkenin update( ) fonksiyonu LeftSideButton sınıfının içerisinde tanımlanmış olan update( ) fonksiyonudur. Bu fonksiyon içerisinde öncelikle, nesnenin selected özelliğinin 'true' ya da 'false' olup olmadığını sınar. Bu özellik 'true' ise, yapımcı yöntem içerisinde tanımlanan ve seçili olma halini vurgulayan kırmızı-220, yeşil-30, mavi-60 değerlerinin oluşturduğu renk dolgu rengi olarak belirlenir. Eğer, selected özelliği false ise, gri-200 değerini taşıyan ve seçili olmama halini vurgulayan renk dolgu rengi olarak belirlenir. Dolgu renginin belirlenmesinden sonra çizgi kalınlığının değeri belirlenir. Bu değer, strokeWeight( ) fonksiyonun parametre değerine 2 girilerek yapılır. Ardından, belirlenmiş dolgu ve çizgi kalınlığı ile nesnenin yaratılması sırasında kullanılan yapımcı yöntemin lokasyon belirleyen özelliklere atadığı yatay ve dikey konum değerleri ile kenar uzunlukları bilgisinin parametrelerini oluşturduğu, processing ortamının standart rect( ) yöntemi çağrılır.

- portamento.update( ): Kullanıcı arayüzünde bulunan bir adet PortamentoTool nesnesini taşıyan portamento değişkeninin update( ) çalıştırılır. portamento değişkeninin update( ) fonksiyonu, synthStore dizisinin güncel elemanının sahip olduğu, yani ızgaralı ekranda seçili olan görüntü kanalının yatay ve dikey sıra bilgisini taşıyan değerlerin işaret ettiği synthStore dizisi elemanının sahip olduğu, portamentoValueData değişkeninin değerini giriş parametresi olarak alır. Burada önemli olan nokta, otuz iki kanal ses dalgasının bütün parametre niceliklerini taşıyan Synth sınıfı türündeki synthStore dizisinin hangi elemanının, yani hangi ses kanalının, ilgili anda arayüzde gösterileceğinin bilinmesidir. Güncel ses kanalı için, ses kanalının bütün bilgileri arayüz nesnelerine atanır. (Şekil 3.29)



Şekil 3.29 portamento değişkeninin update( ) yönteminin, güncel ses kanalı verisinin giriş parametresi yapılması ile çalıştırılması

portamento değişkeninin sahip olduğu update( ) fonksiyonunda ilk olarak, kullanıcıya PortamentoTool nesnesini kullanarak ses kanallarının sahip olduğu portamento değerini kontrol edebilmeleri için yardımcı olacak olan kılavuz çizgilerinin renk ve kalınlık bilgilerinin değerleri belirlenir. Bu değer, stroke( ) ve strokeWeight( ) standart fonksiyonları kullanılarak yapılır. stroke( ) fonksiyonunun parametre değerine 0 girilerek çizgi rengi olarak siyah belirlenir. strokeWeight( ) fonksiyonuna 1 değeri girilir. Dolgu rengi gri-200 değerinde belirlenir. Ardından line( ) standart fonksiyonu ile kılavuz çizgileri çizilir.

Kulpun yatay konumunun, güncel değere göre ne olacağını belirlemek için update( ) fonksiyonuna giriş parametresi olarak alınan değer kullanılır. Bu değerin kullanılmasıyla, computePortamentoValueX( ) adlı sınıf yöntemi çağırılarak kulpun yatay konum değeri hesaplanır. processing ortamının standart rect( ) fonksiyonu ile kulp arayüze çizilir.

- gridNumberSelections[ ].update( ): gridNumberSelections dizisi ses sinyal kanal sayısını belirlemek üzere (bir, dört ya da on altı) kullanılan üç adet LeftSideButton nesnesinden oluşturulmuştur. Bütün gridNumberSelections dizisi elemanlarının update( ) fonksiyonu çalıştırılır. gridNumberSelections dizisi LeftSideButton nesnelerini tuttuğu için, bu update( ) fonksiyonu LeftSideButton sınıfına ait olan bir fonksiyondur. Bu fonksiyon içerisinde öncelikle, nesnenin selected özelliğinin 'true' ya da 'false' olup olmadığını sınar. Bu özellik 'true' ise, yapımcı yöntem içerisinde tanımlanan ve seçili olma halini vurgulayan kırmızı-220, yeşil-30, mavi-60 değerlerinin oluşturduğu renk dolgu rengi olarak belirlenir. Eğer, selected özelliği false ise, gri-200 değerini taşıyan ve seçili olmama halini vurgulayan renk dolgu rengi olarak belirlenir. Dolgu renginin belirlenmesinden sonra çizgi kalınlığının değeri belirlenir. Bu değer, strokeWeight( ) fonksiyonun parametre değerine 2 girilerek yapılır. Ardından, belirlenmiş dolgu ve çizgi kalınlığı ile nesnenin yaratılması sırasında kullanılan yapımcı yöntemin lokasyon belirleyen özelliklere atadığı yatay ve dikey konum değerleri ile kenar uzunlukları bilgisinin parametrelerini oluşturduğu, processing ortamının standart rect( ) yöntemi çağrılır.
- signalNumberSelections[ ].update( ): signalNumberSelections dizisi her bir görüntü çerçevesinden oluşturulan iki ses sinyal kanalından hangisinin güncel seçim dahilinde olduğunun bilgisini tutan iki adet LeftSideButton nesnesinden oluşturulmuştur. Bütün signalNumberSelections dizisi elemanlarının update( ) fonksiyonu çalıştırılır. signalNumberSelections dizisi LeftSideButton nesnelerini tuttuğu için, bu update( ) fonksiyonu LeftSideButton sınıfına ait olan bir fonksiyondur. Bu fonksiyon içerisinde öncelikle, nesnenin selected özelliğinin 'true' ya da 'false' olup olmadığını sınar. Bu özellik 'true' ise, yapımcı yöntem içerisinde tanımlanan ve seçili olma halini vurgulayan kırmızı-220, yeşil-30, mavi-60 değerlerinin oluşturduğu renk dolgu rengi olarak belirlenir. Eğer,

selected özelliği false ise, gri-200 değerini taşıyan ve seçili olmama halini vurgulayan renk dolgu rengi olarak belirlenir. Dolgu renginin belirlenmesinden sonra çizgi kalınlığının değeri belirlenir. Bu değer, strokeWeight( ) fonksiyonun parametre değerine 2 girilerek yapılır. Ardından, belirlenmiş dolgu ve çizgi kalınlığı ile nesnenin yaratılması sırasında kullanılan yapımcı yöntemin lokasyon belirleyen özelliklere atadığı yatay ve dikey konum değerleri ile kenar uzunlukları bilgisinin parametrelerini oluşturduğu, processing ortamının standart rect( ) yöntemi çağrılır.

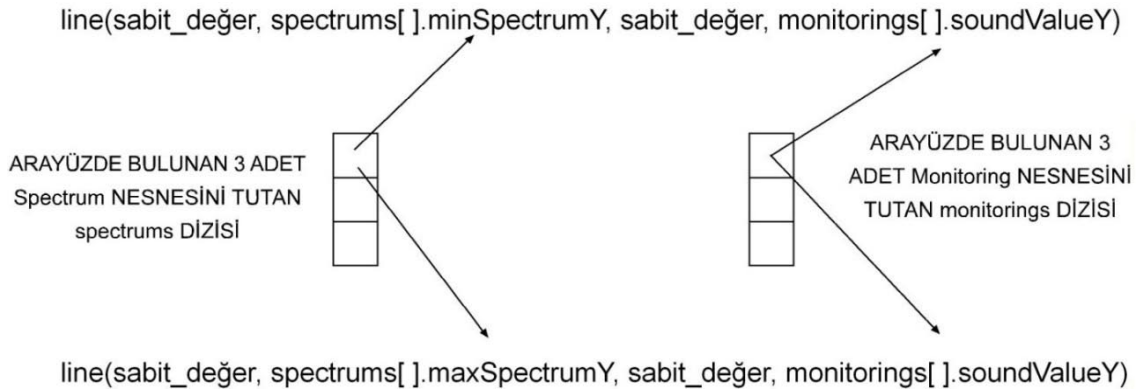
- previewbutton.update( ): Bu değişken, kameradan alınan görüntünün ön izlemesinin yapılıp yapılmayacağının bilgisini taşıyan arayüz nesnesi olmak üzere LeftSideButton sınıfından üretilmiştir. Bu değişkenin update( ) fonksiyonu LeftSideButton sınıfının içerisinde tanımlanmış olan update( ) fonksiyonudur. Bu fonksiyon içerisinde öncelikle, nesnenin selected özelliğinin 'true' ya da 'false' olup olmadığını sınar. Bu özellik 'true' ise, yapımcı yöntem içerisinde tanımlanan ve seçili olma halini vurgulayan kırmızı-220, yeşil-30, mavi-60 değerlerinin oluşturduğu renk dolgu rengi olarak belirlenir. Eğer, selected özelliği false ise, gri-200 değerini taşıyan ve seçili olmama halini vurgulayan renk dolgu rengi olarak belirlenir. Dolgu renginin belirlenmesinden sonra çizgi kalınlığının değeri belirlenir. Bu değer, strokeWeight( ) fonksiyonun parametre değerine 2 girilerek yapılır. Ardından, belirlenmiş dolgu ve çizgi kalınlığı ile nesnenin yaratılması sırasında kullanılan yapımcı yöntemin lokasyon belirleyen özelliklere atadığı yatay ve dikey konum değerleri ile kenar uzunlukları bilgisinin parametrelerini oluşturduğu, processing ortamının standart rect( ) yöntemi çağrılır.
- gridbutton.update( ): Bu değişken, kameradan alınan görüntünün ön izlemesinin yapıldığı alanda bu görüntünün alt çerçevelerinin kılavuz çizgileri ile ayrıştırılıp ayrıştırılmayacağının bilgisini taşıyan arayüz nesnesi olmak üzere LeftSideButton sınıfından üretilmiştir. Bu değişkenin update( ) fonksiyonu LeftSideButton sınıfının içerisinde tanımlanmış olan update( ) fonksiyonudur. Bu fonksiyon içerisinde öncelikle, nesnenin selected özelliğinin 'true' ya da 'false' olup olmadığını sınar. Bu özellik 'true' ise, yapımcı yöntem içerisinde tanımlanan ve seçili olma halini vurgulayan kırmızı-220, yeşil-30, mavi-60

değerlerinin oluşturduğu renk dolgu rengi olarak belirlenir. Eğer, selected özelliği false ise, gri-200 değerini taşıyan ve seçili olmama halini vurgulayan renk dolgu rengi olarak belirlenir. Dolgu renginin belirlenmesinden sonra çizgi kalınlığının değeri belirlenir. Bu değer, strokeWeight( ) fonksiyonun parametre değerine 2 girilerek yapılır. Ardından, belirlenmiş dolgu ve çizgi kalınlığı ile nesnenin yaratılması sırasında kullanılan yapımcı yöntemin lokasyon belirleyen özelliklere atadığı yatay ve dikey konum değerleri ile kenar uzunlukları bilgisinin parametrelerini oluşturduğu, processing ortamının standart rect( ) yöntemi çağrılır.

- linebutton.update( ): Bu değişken, kameradan alınan görüntünün kaynaklık ettiği ses sinyal özellikleri ile kamera değerlerini gösteren arayüz nesnelerinin ilişkilerinin daha rahat görülebilmesi için, aralarına çizilen çizgilerin gösterilip gösterilmeyeceğinin bilgisini taşıyan arayüz nesnesi olmak üzere LeftSideButton sınıfından üretilmiştir. Bu değişkenin update( ) fonksiyonu LeftSideButton sınıfının içerisinde tanımlanmış olan update( ) fonksiyonudur. Bu fonksiyon içerisinde öncelikle, nesnenin selected özelliğinin 'true' ya da 'false' olup olmadığını sınar. Bu özellik 'true' ise, yapımcı yöntem içerisinde tanımlanan ve seçili olma halini vurgulayan kırmızı-220, yeşil-30, mavi-60 değerlerinin oluşturduğu renk dolgu rengi olarak belirlenir. Eğer, selected özelliği false ise, gri-200 değerini taşıyan ve seçili olmama halini vurgulayan renk dolgu rengi olarak belirlenir. Dolgu renginin belirlenmesinden sonra çizgi kalınlığının değeri belirlenir. Bu değer, strokeWeight( ) fonksiyonun parametre değerine 2 girilerek yapılır. Ardından, belirlenmiş dolgu ve çizgi kalınlığı ile nesnenin yaratılması sırasında kullanılan yapımcı yöntemin lokasyon belirleyen özelliklere atadığı yatay ve dikey konum değerleri ile kenar uzunlukları bilgisinin parametrelerini oluşturduğu, processing ortamının standart rect( ) yöntemi çağrılır.

Kullanıcı arayüzünü oluşturan nesnelerin update( ) fonksiyonları çalıştırıldıktan sonra, kullanıcının Spectrum sınıfı nesneleri ile Monitoring sınıfı nesneleri arasında ve Monitoring sınıfı nesneleri ile ActiveButtonHandle sınıfı nesneleri arasında bulunan ilişkiyi daha açık görebilmesi için bu nesneleri birbirine bağlayan çizgilerin çizilmesi işlemi gerçekleştirilir. Bu işlem, linebutton değişkeninin selected özelliğinin 'true' ya da

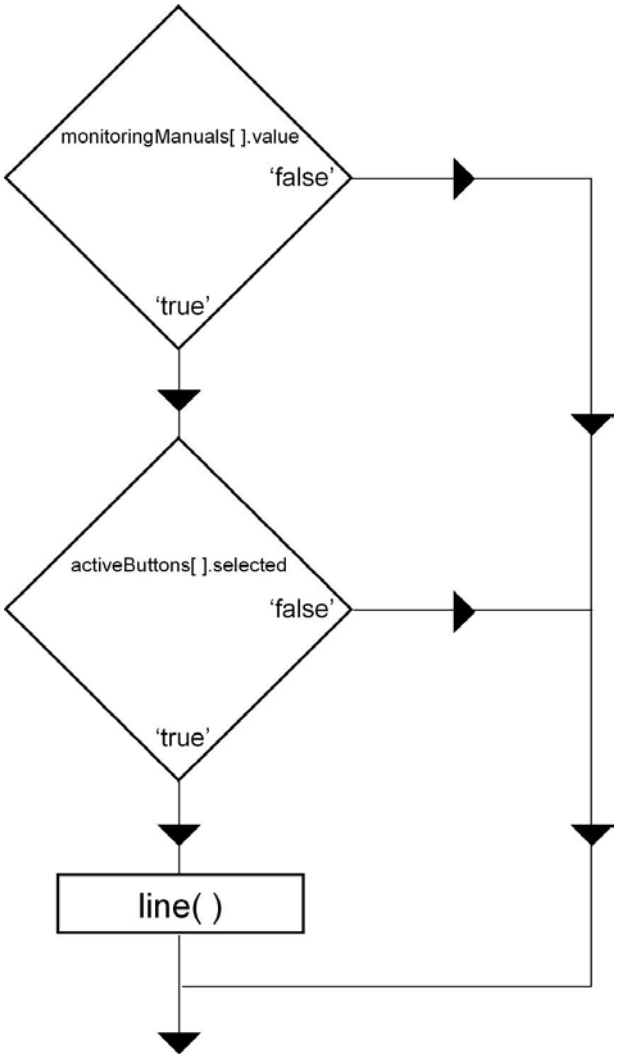
‘false’ olup olmadığının sınanmasının ardından gerçekleştirilir. Bu değişkenin selected özelliği kullanıcı tarafından ‘false’ olarak belirlenmiş ise, ilgili çizgiler çizilmezler. Eğer linebutton özelliği kullanıcı tarafından ‘true’ olarak belirlenmiş ise çizgilerin çizim işlemleri yapılır. Bu çizgilerin ekrana çizilmesi için ilk yapılan standart stroke( ) fonksiyonu ve strokeWeight( ) ile çizgi renginin ve çizgi kalınlığının belirlenmesidir. Çizgi rengi için gri-240 ve çizgi kalınlığı için 1 değeri girilir. Ardından Spectrum nesneleri ile Monitoring nesneleri arasında ilgili çizgiler çizilir. Bunun için; spectrums dizisinin üç elemanının, üst sınırı ve alt sınırı belirleyen kulplarının dikey konum değerini tutan minSpectrumY ve maxSpectrumY özellikleri ve monitorings dizisinin üç elemanının, değer gösteren kulpların dikey konum değerini tutan soundValueY özellikleri kullanılır. Bu kulpların arayüz içerisinde sadece dikey konumları dinamik olarak değiştiği için, yatay konum bilgileri bu nesnelerin yapımcı fonksiyonları çalıştırılırken belirlenen sabit değerlerdir. Bu değerlerin giriş parametrelerini oluşturduğu standart line( ) fonksiyonu spectrums ve monitorings dizilerinin her bir elemanı için iki kere çalıştırılır. Bunlardan ilki, alt sınırı belirleyen spektrum kulpu ile monitoring kulpu arasında, ikincisi ise üst sınırı belirleyen spektrum kulpu ile monitoring kulpu arasında bulunan çizgi için çalıştırılır. (Şekil 3.30)



Şekil 3.30 Spectrum nesnelerinin alt sınır ve üst sınır bilgisini gösteren kulpları ile Monitoring nesnelerinin kulpları arasında çizgi çizilmesi için gerekli olan parametreler

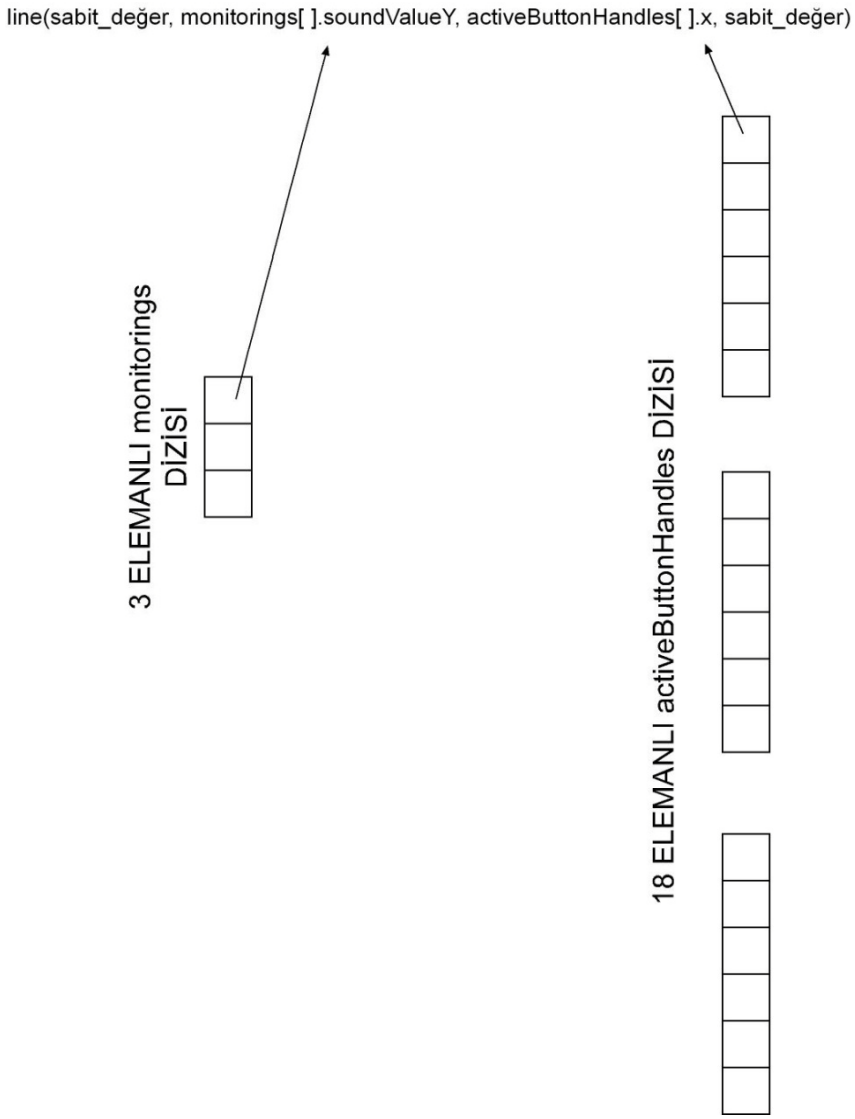
Bu çizimlerin ardından, Monitoring nesnelerinin kulpları ile ActiveButtonHandles nesnelerinin kulpları arasında bulunacak olan çizgiler çizilir. Bu çizimler için öncelikle iki adet sınama yapılır. Bu sınamalardan ilki, ilgili Monitoring nesnesi ile bağlantılı olan MonitoringManual nesnelerinin value özelliklerinin ‘true’ ya da ‘false’ olmasını sınar.

Eğer kullanıcı, güncel ses sinyal kanalına ait ilgili ses özelliğinin değerini kamera ile belirlemek istemiyor ise ve bunun doğrultusunda monitoringManuals dizisinin ilgili elemanının value değeri ‘false’ ise bu çizgiler çizilmeyecektir. Diğer sınama, ilk sınamanın ‘true’ olması doğrultusunda anlam kazanır. Burada da, ilgili Monitoring nesnesinin, kameradan gelen altı adet parametreden hangileri ile ilişkiye gireceğinin sınamasıdır. Bu parametrelerin kullanıcı kontrolü ile ses sinyallerine değer aktaracağı düşünüldüğünde, her ActiveButton nesnesi için çizgi çizilmemesi gerektiği görülür. Bu yüzden, hangi ActiveButtonHandle nesnesi için Monitoring nesnesi ile arasında çizgi gerekeceğine sınamayla karar verilir. Bu sınamalar arayüzde bulunan on sekiz adet ActiveButtonHandle nesnesi için ayrı ayrı yapılır. (Şekil 3.31)



Şekil 3.31 Monitoring nesneleri ile ActiveButtonHandle nesneleri arasında çizilecek olan çizgilerin öncesindeki sınamaların akış diyagramı

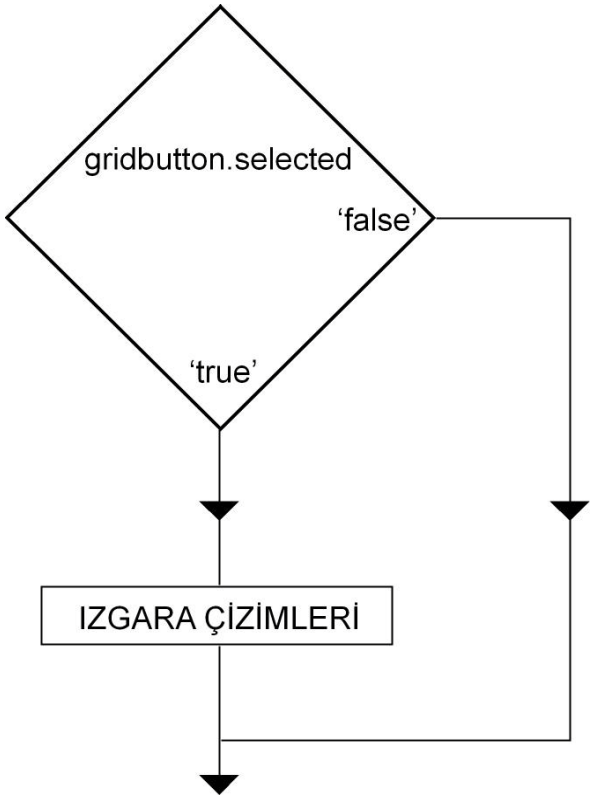
Her bir ActiveButtonHandle için yapılan sınamalardan sonra, ilgili kulplar arasındaki çizgilerin çizimleri yapılır. Bu çizim standart line( ) fonksiyonu ile yapılır. (Şekil 3.31) Bu fonksiyonun parametreleri; monitorings dizisinin ilgili elemanının kulpun taşıdığı değere göre arayüzde bulunacağı dikey lokasyon bilgisini taşıyan soundValueY özelliği, activeButtonHandles dizisinin ilgili elemanının kulpun taşıdığı değere göre arayüzde bulunacağı yatay lokasyon bilgisini taşıyan x özelliğidir. Diğer parametreler, Monitoring nesneleri üretilirken kullanılan yapımcı yöntemlerin içerisinde belirlenen sabit yatay konum bilgisi ve ActiveButtonHandle nesneleri üretilirken kullanılan yapımcı fonksiyonların içerisinde belirlenen her bir nesnenin sabit dikey konum bilgisidir. Bu parametre yapısı ile standart line( ) fonksiyonu, sınamalardan geçmiş bütün ActiveButtonHandle nesnesi ile Monitoring nesnesi arasında çizgi çizmek için kullanılır. (Şekil 3.32)



Şekil 3.32 ActiveButtonHandle ve Monitoring nesneleri arasında bulunan çizgilerin çizilmesi için gerekli olan parametreler

Kullanıcı arayüzünde bulunan ve kamera kaynaklı parametreler ile bunlara bağlı olarak belirlenen ses sinyal kanalı özelliklerinin gösterildiği kulplar arasında gösterilecek olan çizgilerin çizilmesinin ardından, ön izleme ekranında kullanıcının alt çerçeveleri görmesini sağlayan ızgara yapısının çizimine geçilir. Bu ızgara çizgilerinin çiziminden önce de, bir sınama yapılır. Bu sınama, gridbutton değişkeninin selected özelliğini sınar. Bu değişkenin selected özelliği, eğer kullanıcı ön izleme ekranını ızgaralı olarak kullanmak istemeyecekse ‘false’ değerini taşır. Bu durumda herhangi bir ızgara çizgisi çizilmez. Eğer, kullanıcı ızgara çizgilerini görmek istiyor ise, bu durumda gridbutton

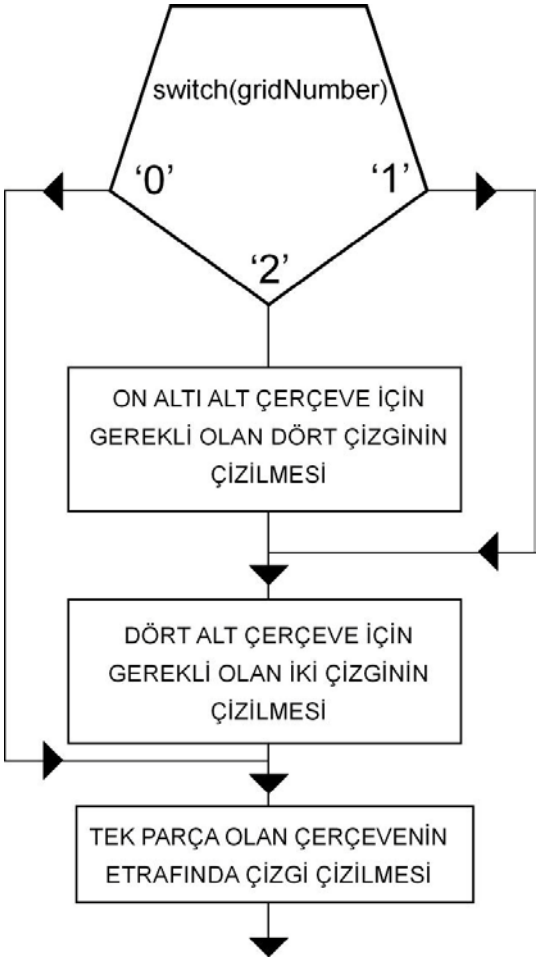
değişkeninin selected özelliği ‘true’ değerini taşıyacaktır. Bu durumda ızgara çizgileri çizilir. (Şekil 3.33)



Şekil 3.33 Izgaram çiziminin gridbutton değişkeninin selected özelliğinin ‘true’ ya da ‘false’ olmasına göre belirlenmesinin akış diyagramı

Izgaraların çizilmesi için ikinci bir sınama gerekir. Kullanıcının ön izleme ekranını kaç alt çerçeveye böldüğünün bilgisi bu çizimler için gerekli dallanmayı sağlayacaktır. Kullanıcı arayüzünde bulunan ve kamera kaynağından gelen görüntünün kaç alt çerçeveye bölüneceğinin (bir, dört ya da on altı) bilgisini taşıyan gridNumberSelections dizisine ait nesneler ile etkileşim kurarak niceliği belirlenen gridNumber global değişkeninin taşıdığı değere göre dallanmalar gerçekleştirilir. Eğer kullanıcı alt çerçevelerin olmayacağı tek parçalı bir görüntüleme seçtiyse (bu durumda gridNumber değişkeni 0 değerini taşır), sadece ön izleme ekranının etrafına bir çerçeve çizgisi çizilecek demektir. Kullanıcı dört alt çerçevenin olacağı bir görüntüleme seçtiyse (bu durumda gridNumber değişkeni 1 değerini taşır), ön izleme ekranının etrafına çizilecek olan çerçeve dışında, ekranı dört eşit parçaya bölen yatay ve dikey olmak üzere birer ızgara çizisi daha çizilecektir. Kullanıcı on altı alt çerçevenin olacağı bir görüntüleme

seçtiyse (bu durumda gridNumber değişkeni 2 değerini taşır), ön izleme ekranının etrafına çizilecek olan çerçeve ve ekranı dört eşit parçaya bölen yatay ve dikey olmak üzere birer ızgara çizisi dışında, görüntüyü on altı eşit parçaya bölecek dört çizgi daha çizilecektir. (Şekil 3.34)



Şekil 3.34 Kullanıcının ön izleme ekranının kaç alt çerçeveye bölüneceğini seçmesi ile belirlenen ızgara çizimlerinin akış diyagramı

Izgara çizgilerinin renk değeri olarak kırmızı-200, yeşil-0, mavi-0 değerleri seçildi. Çizgi kalınlığı olarak da, strokeWeight( ) fonksiyonuna 2 değeri girildi. Ardından çizgi çizilmesi için standart line( ) fonksiyonu kullanıldı. Sadece, ön izleme ekranının etrafına çizilecek olan çerçeve için gereken rect( ) fonksiyonu öncesi strokeWeight( ) fonksiyonuna 1 değeri girildi.

GraphicUserInterface( ) fonksiyonunun kullanıcı arayüzünün oluşturulması için yerine getireceği son işlev; kameradan gelen görüntünün parçalandığı alt çerçevelerin her birinin bir ses kaynağı olmasından dolayı, hangi görüntü kaynağının güncel olarak arayüzde gösterileceğinin seçiminin yapılacağı aracı görsel olarak oluşturmaktır.

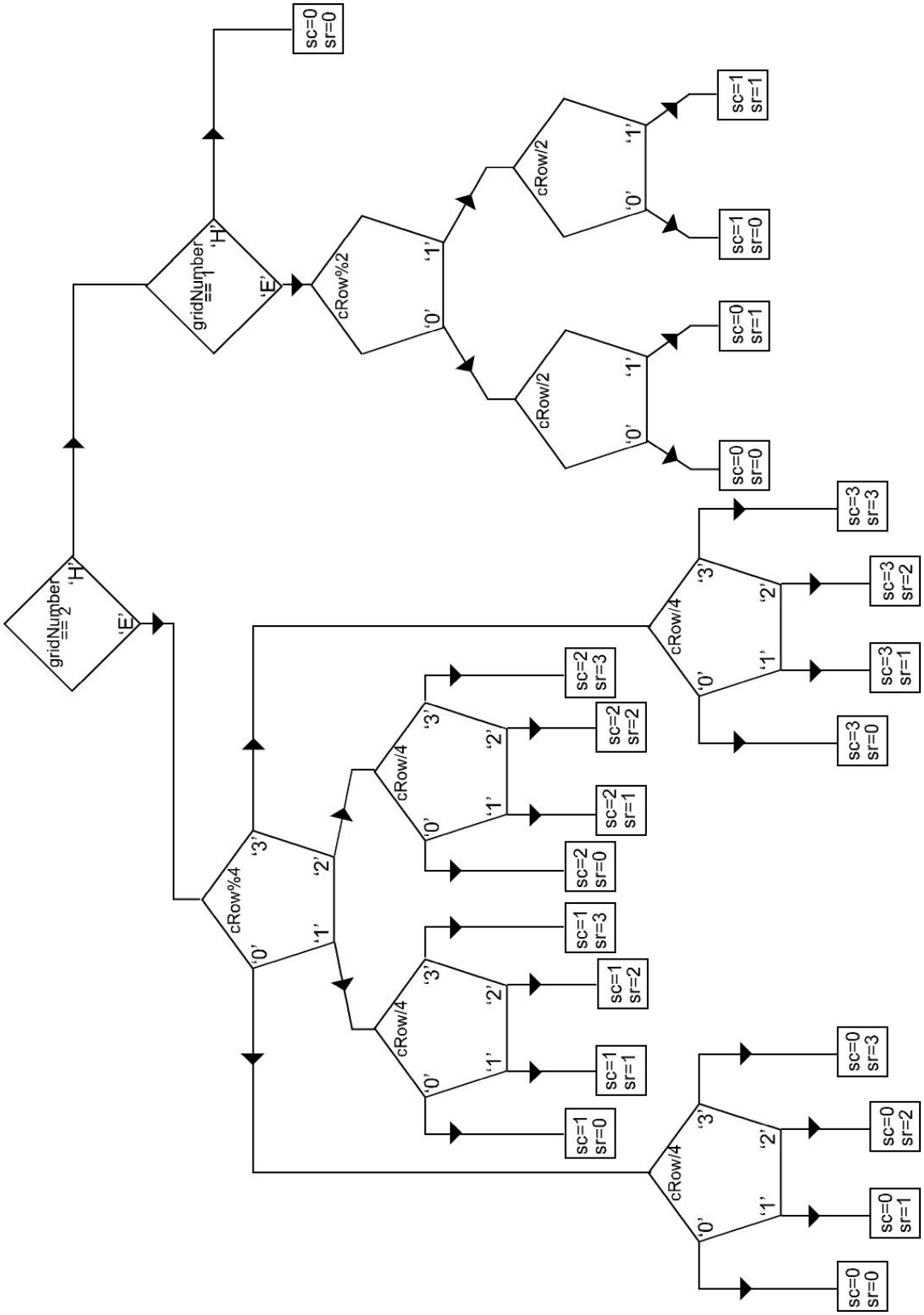
Bunun için ilk olarak, bu seçim alanının dış hattının oluşturulması için gereken çerçevenin çizilmesidir. Bu çerçeveyi oluşturacak olan çizginin renk değerinin siyah olması için stroke( ) fonksiyonuna gri-0 değeri ve çizgi kalınlığı için, strokeWeight( ) fonksiyonuna 2 değeri parametre olarak atanır. Ardından, bu seçim arayüz elemanının iç ızgara çizimlerinin yapılması için, öncelikle noFill( ) komutu ile dolgusuz bir çizgi tarzı seçilir. Çizgi rengi olarak stroke( ) fonksiyonuna kırmızı-200, yeşil-0, mavi-0 değerleri atanır. Çizgi kalınlığı olarak strokeWeight( ) fonksiyonuna 2 değeri atanır. Bunun ardından seçim alanının ızgara çizgileri ile kaplanması için ön izleme ekranında ızgara çizgileri oluşturmak için uygulanan iş akışının benzeri uygulanır. Burada da, iş akışının dallanması için gridNumber global değişkeni kullanılır. (Şekil 3.34)

Seçim alanının dış çerçevesi ve iç ızgaralarının çizilmesinin ardından, güncel olan ızgara alanının boyanarak seçili olması sağlanır. Bunun için iç içe if( ) ve switch( ) sınamaları yapılır. Bu sınamalarda seçili olan ızgara alanının hangi sıra ve sütunda olduğunun bilgisini tutacak iki adet yerel integer temel tipinde değişken tanımlanır. Bunlardan adı sr olan değişken, seçili olan ızgara alanının sıra değerini; adı sc olan değişken, seçili olan ızgara alanının sütun değerini taşır. Bu değerler, hem sıra hem de sütun için 0, 1, 2, 3 değerlerinden birini taşıyabilirler. Bu yapı içerisinde önemli olan, global olarak tanımladığımız, ilk değerini 0 olarak alan ve mouseReleased( ) standart olayı içerisinde her defasında yeniden güncellenen cRow değişkeninin değeridir. cRow değişkeni, çalışmada daha sonra anlatılan mouseReleased( ) standart olayı içerisinde bu ızgaralı alanda yapılan seçime göre yeniden güncellenir. Bu değişken sürekli olarak kullandığımız matris yapılı synthStore dizisinin sütun değerini belirleyen değişkendir. Böylece kullanıcı arayüzünde hangi görüntü kanalının kaynaklık ettiği ses kanalı özelliklerini görmekte olduğumuz bilgisi sürekli olarak güncel tutulur. Çalışmada daha önce anlatıldığı gibi, cRow değişkeni 0 ile 15 değerleri arasında değişebilir. Bu da birinci ve on altıncı görüntü kanalını işaret edebildiğini göstermektedir.

Bu yapı içerisinde ilk yapılan sınama, gridNumber değişkeninin taşıdığı değerdir. Bu değer eğer 2 sayısına eşitse, kullanıcı on altı alt çerçeve sayısını seçmiştir. Bu durumda, sc ve sr değişkenleri 0, 1, 2, 3 değerlerinden herhangi birini taşıyabilirler. Eğer, cRow değişkeninin taşıdığı değerin 4e bölünmesinden sonra kalan 0 ise sc değeri 0 olarak atanır. Eğer, bu bölme işleminden kalan değer 1 ise sc değeri 1 olarak atanır. Eğer, bu bölme işleminden kalan değer 2 ise sc değeri 2 olarak atanır. Eğer, bu bölme işleminden kalan değer 3 ise sc değeri 3 olarak atanır. cRow değişkeninin 4 e bölümünde bölüm değeri 0 ise sr değeri 0 atanır. Eğer bölüm değeri 1 ise sr değeri 1 atanır. Eğer bölüm değeri 2 ise sr değeri 2 atanır. Eğer bölüm değeri 3 ise sr değeri 3 atanır.

Kullanıcı dört alt çerçeve sayısını seçmiş ise gridNumber değişkeninin taşıdığı değer 1 olacaktır. Bu durumda ilk yapılan if( ) sınaması 'false' olacak ve ikinci if( ) sınamasına geçilecektir. Bu sınamada eğer gridNumber değeri 2 ise cRow değeri 2'ye bölünür. Bu bölümde kalan 0 ise, sc değerine 0 atanır. Eğer bu bölümde kalan 1 olursa, sc değerine 1 değeri atanır. cRow değerinin 2'ye bölünmesi sonucunda oluşan bölüm değeri 0 ise sr değişkenine 0 değeri atanır. Bölüm değeri 1 ise sr değişkenine 1 değeri atanır.

Kullancı tek çerçeveli görüntü seçimini yapmış ise gridNumber değişkeninin taşıdığı değer 0 olacaktır. Bu durumda, ikinci yapılan if( ) sınaması da 'false' değerini alacaktır. Bu durumda sc v sr değişkenlerine direk olarak 0 değeri atanır. Çünkü kullanıcı tek çerçeveli bir görüntü kanalı ile çalışacaktır ve bu durumda görüntü kaynağının değiştirilme ihtimali yoktur. Bütün bu sınamalardan sonra, kullanıcının ızgaralı yapıda hangi satır ve sütundaki hücre ile güncel olarak çalıştığı belirlenmiştir. (Şekil 3.35)



Şekil 3.35 Kullanıcının ızgaralı görüntü alanında hangi satır ve sütundaki hücre ile güncel olarak çalıştığının bilgisinin if( ) ve switch( ) sınamaları ile bulunması

Sırada, ızgaralı olarak bölünmüş olan görüntü kanallarının güncel seçiminin yapılacağı seçim alanında ön izleme ekranının bir benzeri olarak oluşturulmuş olan ızgaralı yapının içindeki hücrelerden seçili olanın, diğerlerinden ayırt edilmesi için içinin boyanması görevi vardır.

Bunun için ilk yapılması gereken, bu ızgaralı seçim alanındaki her bir hücrenin boyutunun bilinmesidir. Bunun için, daha önce bu seçim alanının dış çerçevesi oluşturulurken kullanılan değer ele alınacaktır. Ancak, bu durumda alt çerçeve sayısının anlaşılması için gridNumber değişkenine geri dönülür. Bu değişkenin değeri 1 ise, toplam dört alt çerçeve var demektir. Bu durumda dış çerçeve uzunluğu yatayda ve dikeyde ikiye bölünecektir. Eğer gridNumber değişkeninin değeri 2 ise toplam on altı alt çerçeve var demektir ve dış çerçeve uzunluğu yatayda ve dikeyde dörde bölünecektir. Buradan çıkan sonuç s yerel değişkeninde tutulacaktır. Kare olan ızgaralı alanın her hücresinin de kare olacağı düşünüldüğünde, güncel hücrenin her bir ayrıtının uzunluğunu yukarıdaki sınamadan sonra s yerel değişkeninde tutuluyor olacaktır.

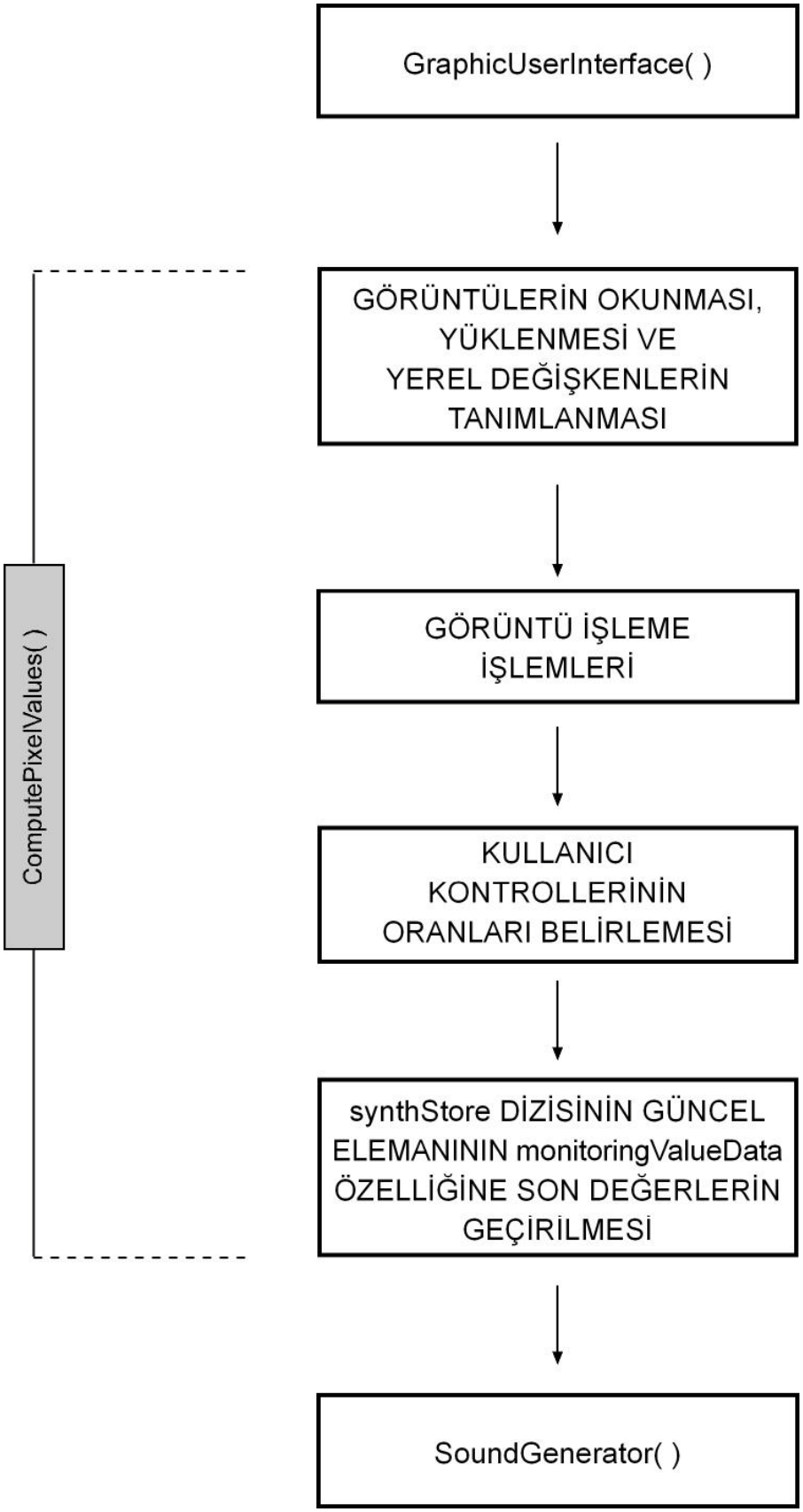
Güncel hücrenin çizilmesi için dolgu rengi olarak kırmızı-200, yeşil-0, mavi-0 ve alfa-100 değerleri belirlenir. Hücreye çizilecek olan kırmızı alanın, ızgaralı seçim alanında konumlandırılacağı yerin yatay bilgisi, s*sc işlemi ile belirlenir. Bu kırmızı alanın dikey konum bilgisi ise, s*sr işlemi ile belirlenir. Bu belirlemenin ardından, yatay ve dikey konum bilgileri ile kenar uzunluklarının (s değişkeni tutmaktadır) parametre olarak girdiği rect( ) fonksiyonu çağırılır. Böylece ızgaralı seçim alanında güncel olduğu belli edilecek olan hücre kırmızıya boyanarak diğerlerinden ayrılır.

Böylece kullanıcı grafik arayüzünün yaratılması görevi için oluşturulmuş olan GraphicUserInterface( ) fonksiyonu sonlanır. Bu fonksiyon, draw( ) döngüsü içerisinde, bir önceki döngünün sonunda oluşan görüntü ve ses parametrelerine göre kullanıcı arayüzünü yeni döngünün başında güncelleme görevi ile ilk fonksiyon olarak çalıştırılır.

3.3.4 Görüntü Yakalayan Donanımdan Alınan Görüntülerin İşlenmesi ve Ses Sinyal Parametrelerine Dönüştürülmesi

Yazılımda draw( ) döngüsünün içerisinde bulunan GraphicUserInterface( ) fonksiyonun ardından, ikinci olarak ComputePixelValues( ) fonksiyonu çalıştırılır. Bu fonksiyonun öncelikli amacı, kameradan alınan her bir görüntü karesinin sahip olduğu piksel değerlerini ölçmek ve bu değerleri var olan alt çerçevelerin her biri için ayrı ayrı tutacak olan değişkenlerde saklamaktır. ComputePixelValues( ) fonksiyonu, kendisine gelen dijital görüntü verilerini, ses sinyal kanalları için anlamlı verilere çevirmek için dönüştürücü görevini üstlenecek ve bu verilerin kullanıcı kontrolü ile oluşturulmuş olan oranlarda karıştırılmasını ve ses sinyal kanallarının değerlerini tutan değişkenlere bu değerlerin aktarılmasını sağlayacaktır.

ComputePixelValues( ) fonksiyonunun toplamda dönüştürücü olarak yapacağı iş için, öncelikle bu fonksiyon içerisinde işlevsel olarak gerekli olan yerel değişkenler tanımlanacaktır. Ardından görüntü çerçevelerinde oluşan piksel değerlerinin tek tek ele alarak işlenmesi ve bu alanlarda bulunan kırmızı, yeşil, mavi, renk, doygunluk ve parlaklık değerlerinin ortalamaları alınacaktır. Bu değerler, kullanıcının kontrolü ile belirlenen oranlarda karıştırılacak ve ses sinyalleri için gerekli olan değerler üretilecektir. Son olarak üretilen değerler, ses sinyal kanallarının bütün değerlerini tutan synthStore dizisinin ilgili elemanının, kameradan gelen değerleri tutan monitoringValueData dizi özelliğine aktarılacaktır.(Şekil 3.36)

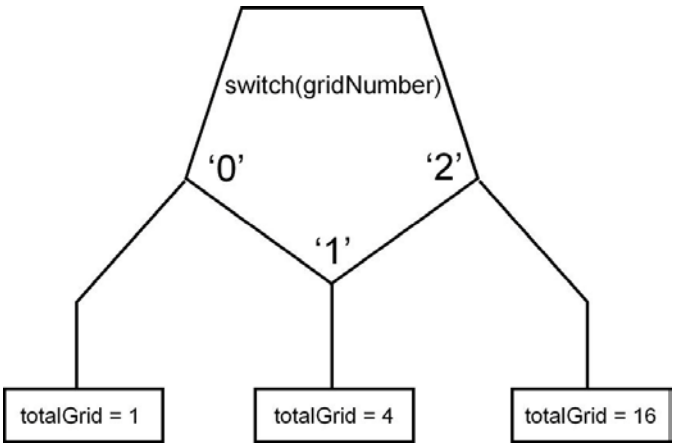


Şekil 3.36 ComputePixelValues( ) fonksiyonunun işleyişi

3.3.4.1 Kameradan görüntü yüklenmesi ve yerel değişkenlerin tanımlanması

Bu görevin yerine getirilmesi için, ilk olarak cam değişkeninde bulunan görüntünün okunması ve bu okunan görüntülerin cam değişkeninin gösterdiği arayüz alanına pikseller olarak yüklenmesi gerekmektedir. Bunlar, processing ortamının processing.video paketi ile bize sağladığı Capture sınıfından üretilen cam değişkenin read( ) ve loadPixels( ) yöntemleri ile gerçekleştirilir. Böylece cam değişkenin içerdiği görüntü karesi kameradan gelen verilere göre güncellenmiş olur.

Kameradan gelen görüntü verilerine göre yapılan güncellemenin ardından, ön izleme ekranında oluşturulan alt çerçeve sayısını tutacak olan integer temel tipinde yerel totalGrid değişkeni tanımlanır. Bu değişken gridNumber global değişkeninin taşıdığı değere göre kendi değerini alır. Global gridNumber değişkeninin değeri 0 ise, yerel totalGrid değişkeninin değeri 1; gridNumber değişkeninin değeri 1 ise, yerel totalGrid değişkeninin değeri 4; gridNumber değişkeninin değeri 2 ise, yerel totalGrid değişkeninin değeri 16 olacaktır. Bu değişken, toplam alt çerçeve sayısının esas niceliğini tutacaktır. (Şekil 3.37)

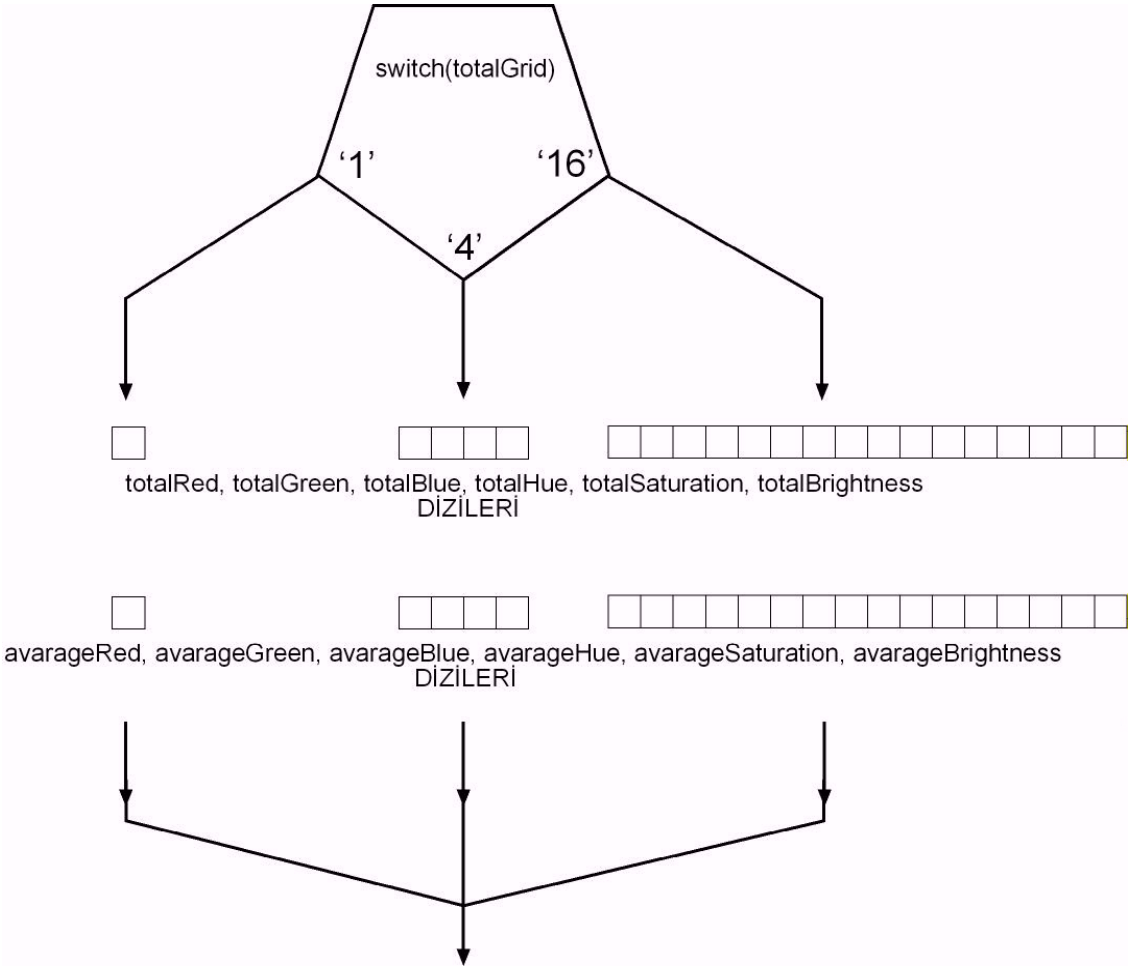


Şekil 3.37 Global gridNumber değişkenine göre, yerel totalGrid değişkenine değer atanması

Ardından, kameradan gelen değerlerin toplanacağı yerel değişkenler tanımlanır. Bu değişkenler, görüntünün ayrıştırılması ve işlenmesi ile elde edilen altı adet özelliğin niceliklerini tutacaktır. Bu yüzden her bir görüntü özelliği için bir değişken tanımlanır. Görüntünün kırmızı değeri için totalRed yerel değişkeni, yeşil değeri için totalGreen

yerel değişkeni, mavi değeri için totalBlue yerel değişkeni, renk değeri için totalHue yerel değişkeni, doygunluk değeri için totalSaturation yerel değişkeni, parlaklık değeri için totalBrightness yerel değişkeni tanımlanır. Ancak bu yapısıyla, bu değişkenler sadece tek bir alt çerçevenin görüntü değerlerini tutabilecek şekildedir. Kullanıcının birden fazla alt çerçeve belirleme durumlarında ise, birden fazla alt çerçevenin değerini tutabilme kabiliyetine sahip olabilmesi için ihtiyaç duyulan bu değişkenler dizi olarak tanımlanmalıdır. Bu dizi değişkenleri tek boyutlu olarak tanımlanacaktır. Dizilerin eleman sayısı ise totalGrid değişkeninin taşıdığı değere göre belirlenecektir. Dolayısı ile bu diziler kullanıcı tek çerçeveli bir görüntü seçti ise tek eleman boyutunda, dört alt çerçeveli bir görüntü seçti ise dört eleman boyutunda, on altı alt çerçeveli bir görüntü seçti ise an altı eleman boyutunda olacaklardır.

Tanımlanması gereken diğer bir değişken grubunun işlevi ise, her bir çerçeve içerisinde bulunan piksellerin altı ayrı niteliğinin toplamını aldıktan sonra, bunların ortalamalarını yani piksel başına düşen değerlerini tutacak olma işidir. Bu işlevi karşılayacak olan değişken grubu da, kullanıcının birden fazla alt çerçeve belirlemesi durumuna karşılık verebilmesi için dizi yerel değişkeni olarak belirlenecektir. Görüntünün kırmızı değerinin ortalaması için averageRed yerel dizi değişkeni, yeşil değerinin ortalaması için averageGreen yerel dizi değişkeni, mavi değerinin ortalaması için averageBlue yerel dizi değişkeni, renk değerinin ortalaması için averageHue yerel dizi değişkeni, doygunluk değerinin ortalaması için averageSaturation yerel dizi değişkeni, parlaklık değerinin ortalaması için averageBrightness yerel dizi değişkeni tanımlanır. Bu diziler de, totalGrid yerel değişkeninin taşıdığı değer kadar eleman taşıyacak boyutta olmalıdır. Böylece toplamı alınan görüntü değerleri, ortalamaları alınarak tutulabilecektir. (Şekil 3.38)

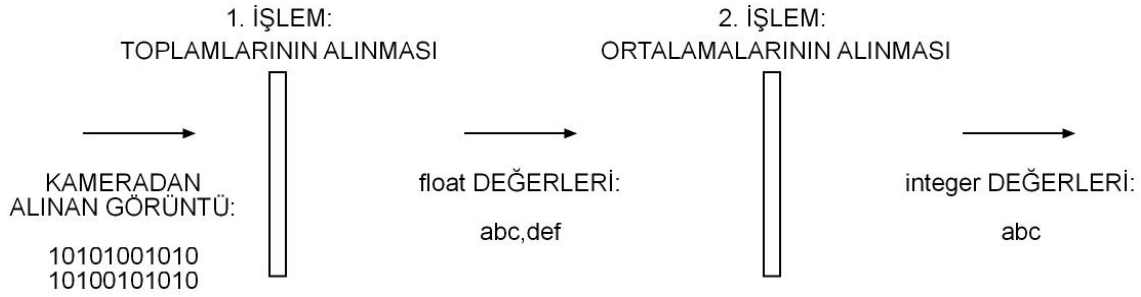


Şekil 3.38 totalGrid yerel değişkenine göre boyutları belirlenen yerel diziler

Piksellerde aranan niteliklerin değerlerinin toplamını tutan totalRed, totalGreen, totalBlue, totalHue, totalSaturation, totalBrightness yerel dizi değişkenlerinin önemli bir özelliği, veri tipi olarak float temel tipinde olmalarıdır. Bu dizilerin, kameradan alınan görüntünün yüklendiği cam değişkeninin değerlerinin ondalıklı olması ve daha sonrasında bu değerlerin ortalama ve oran alma gibi işlemlere tabi tutulacak olmalarından dolayı, toplama işlemi sırasında daha hassas neticeler elde edebilmek amacıyla float temel tipinde olmaları uygun görülmüştür.

totalRed, totalGreen, totalBlue, totalHue, totalSaturation, totalBrightness yerel dizi değişkenlerinde tutulan değerlerin ortalamalarının, yani piksel başına düşen değerlerinin niceliklerinin ise averageRed, averageGreen, averageBlue, averageHue, averageSaturation, averageBrightness dizilerinde tam sayı olarak tutulması uygun görülmüştür. Bu yüzden veri tipi olarak integer temel tipinde tanımlanmışlardır. Bu

dizilerin integer temel tipinde tanımlanmalarının nedeni ise, buradaki değerlerin kullanıcı kontrollerine bağlanması dolayımı ile ses sinyal değerlerine tam ve kesin sayılar olarak verilmeleri içindir. (Şekil 3.39)



Şekil 3.39 Dönüştürme işlemleri sırasında değişen veri tipleri

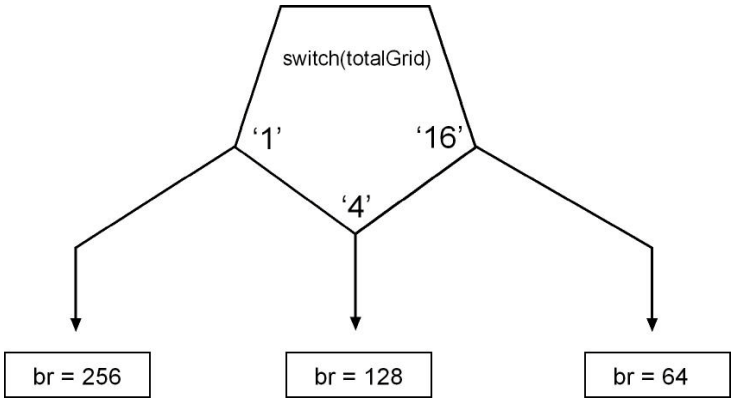
Burada tanımladığımız on iki dizi değişkeninin yerel olmalarının sebebi ise, kullanıcının sürekli olarak alt çerçeve sayısını değiştirerek gridNumber, dolayısı ile totalGrid değişkeninin değerini değiştirebilmesidir. Eğer farklı çerçeve sayıları için farklı boyutlardaki değişkenleri global olarak tanımlarsak, sistem kaynaklarını özellikle de sistem belleğini gereksiz yere meşgul etmiş olacağız. ComputePixelValues( ) fonksiyonunun her çalıştırılmasında, bu dizi değişkenlerinin yerel olarak ve güncel kullanıcı seçimine göre gerekli boyutta yaratılması ve bu fonksiyon bittiğinde de bellekten silinmesi daha doğru bir çözüm olacaktır.

Bu dizi değişkenlerinin kullanıma hazırlanması için, bütün elemanlarının ilk değerlerinin verilmesi gerekmektedir. İlk değer, totalRed, totalGreen, totalBlue, totalHue, totalSaturation, totalBrightness yerel dizi değişkenlerinin bütün elemanları için 0,0 olacaktır. Diğer averageRed, averageGreen, averageBlue, averageHue, averageSaturation, averageBrightness yerel dizi değişkenlerinin bütün elemanları için de 0 olacaktır. Görüntü işleme için, her gelen yeni görüntüde piksellerin ilgili değerleri toplanacağından ve ardından ortalamaları alınacağından dolayı başlangıçta sıfırlanmaları gerekmektedir.

3.3.4.2 Görüntülerin işlenerek kullanıcı kontrolü için parametrelere dönüştürülmesi

Kameradan alınan görüntülerin okunması, yüklenmesi ve ardından yapılan yerel değişkenlerin tanımlanması işlemlerinden sonra, kameradan alınan bu görüntülerin işlenerek kullanıcı kontrolüne verilmek üzere anlamlı parametrelere dönüştürülmesi işlemine geçilir.

Öncelikle, kullanıcının kararı ile belirlenen alt çerçeve sayısına bağlı olarak her bir görüntü alanının kaç piksel uzunluğunda olduğu hesaplanır. Bunun için totalGrid değişkeninde tutulan değerin karekökü alınır. Ardından, bu işlemin sonunda oluşan değer cam değişkeninin width özelliğinin taşıdığı değeri böler. Böylece her bir alt çerçevenin kaç piksel uzunluğunda olduğu bulunur. Bu uzunluk değeri, integer temel tipinde değer taşıyan br yerel değişkenine atanır. Kameradan alınan görüntü 256 * 256 piksel boyutlarında ve kare formatında olduğuna göre, alt çerçeveler de yine kare formatında olacaktır. Bu durumda, eğer kullanıcı tek çerçeveli bir görüntü ile çalışmak isterse bu çerçeve 256 * 256 piksel boyutlarında olacaktır. Dört çerçeveli bir görüntü ile çalışmak isterse bu çerçeveler 128 * 128 piksel boyutlarında olacaktır. On altı çerçeveli bir görüntü ile çalışmak isterse bu çerçeveler 64 * 64 piksel boyutlarında olacaktır. Kare formatında olan bu çerçevelerin, hem yatay hem de dikey uzunluk değerleri birbirine eşit olacağından, tek bir integer temel tipindeki br değişkenin kullanılması uygun olmuştur. (Şekil 3.40)



Şekil 3.40 Kullanıcının kararları doğrultusunda belirlenen totalGrid değişkeni ile alt çerçevelerin uzunluk bilgisinin br değişkenine atanması

Her alt çerçevenin sahip olduğu uzunluk bilgisinin belirlenmesinin ardından, sıra bu alt çerçevelerin içlerindeki piksel değerlerinin toplanmasına gelir. Bu işlem için iç içe döngüler kullanılır. totalGrid değişkeninin karekökü kadar yatayda ve yine totalGrid değişkeninin karekökü kadar dikeyde alt çerçeve olduğuna göre totalGrid değişkeninin karekökü kadar çalışacak iki for döngüsü, toplamda bütün alt çerçeveleri gezebilecektir.

Bütün alt çerçevelerin içlerinde bulunan piksel değerlerinin hesaplanabilmesi için, toplamda alt çerçeve sayısı kadar çalışacak olan içe içe for döngülerinin her çalışmasında ilk olarak, ilgili alt çerçevenin sol üst noktasının referans noktası olarak belirlenmesi işlemi yapılır. Böylece her alt çerçevenin piksel değerlerinin hesaplanması sırasında hangi noktanın sabit olarak kullanılacağı belirlenir. Bunun için, etkinlik alanı bu for döngüsü olan conspoint adında integer temel tipinde yerel bir değişken tanımlanır. Bu değişken içe içe tanımlanan for döngüleri her çalıştığında, yani her alt çerçeveye gelindiğinde yeniden hesaplanır. Bu hesaplama; birinci döngünün indeks değişkeni (i), cam değişkeninin yatay piksel uzunluğu, br değişkeninin değerinin çarpılmasıyla elde edilen değer ile ikinci döngünün indeks değişkeni (j) ve br değişkeninin değerinin çarpılması ile elde edilen değerlerin toplanması ile bulunur. Bu değer conspoint değişkenine atanır. Böylece ilgili alt çerçevenin sol üst köşesinde bulunan pikselin, cam değişkeninde kaçıncı piksel olduğu belirlenmiş olur.

Gelinen alt çerçevenin referans noktasını belirledikten sonra, yatayda ve dikeyde bulunan pikseller için yine iç içe iki for döngüsü tanımlanır. Bu for döngülerinin her biri br değişkeninin taşıdığı değer kadar çalışır. Belirlenen sabit referans noktasının değerinin teker teker arttırılması ile ilgili alt çerçevelerin değerleri okunur. Görüntüyü taşıyan cam değişkeninin pixels[ ] dizi tipindeki özelliği kullanılarak teker teker ilgili alt çerçevenin bütün piksellerine ulaşılır. cam değişkeninin pixels dizi özelliğinin indeks parametresi; conspoint değişkeninin değeri, birinci döngünün indeks değişkeni (a) ile cam değişkeninin width değerinin çarpımından çıkan sonuç ve ikinci döngünün indeks değişkeninin (b) değerinin toplanması ile belirlenir.

İkinci iç içe döngüde, ulaşılan bu piksellerin değerlerinin tutulması için, colorValue isminde ve color tipinde yerel bir değişken tanımlanır. Böylece, incelenecek olan pikselin değerleri artık colorValue değişkeninden ulaşılabilir olur.

Bunların ardından, her alt çerçevenin toplam kırmızı, yeşil, mavi, renk, doygunluk ve parlaklık değerleri için yarattığımız totalRed, totalGreen, totalBlue, totalHue, totalSaturation, totalBrightness yerel dizi değişkenlerinin hangi elemanlarının değerlerine atama yapılacağını belirlemek için güncel indeks değerleri oluşturulur. Bu değerler, birinci iç içe döngünün ilk for döngüsünün indeks değişkeni (i) ile totalGid değişkeninin karekökünün çarpımının ikinci iç içe döngünün ikinci for döngüsünün indeks değişkeni (j) ile toplanması ile oluşturulur.

İlk olarak, totalRed dizisinin ilgili elemanına, red( ) standart fonksiyonunun parametre girişine colorValue değişkeninin girilmesi ile elde edilen kırmızı değeri atanır.

İkinci olarak, totalGreen dizisinin ilgili elemanına, green( ) standart fonksiyonunun parametre girişine colorValue değişkeninin girilmesi ile elde edilen yeşil değeri atanır.

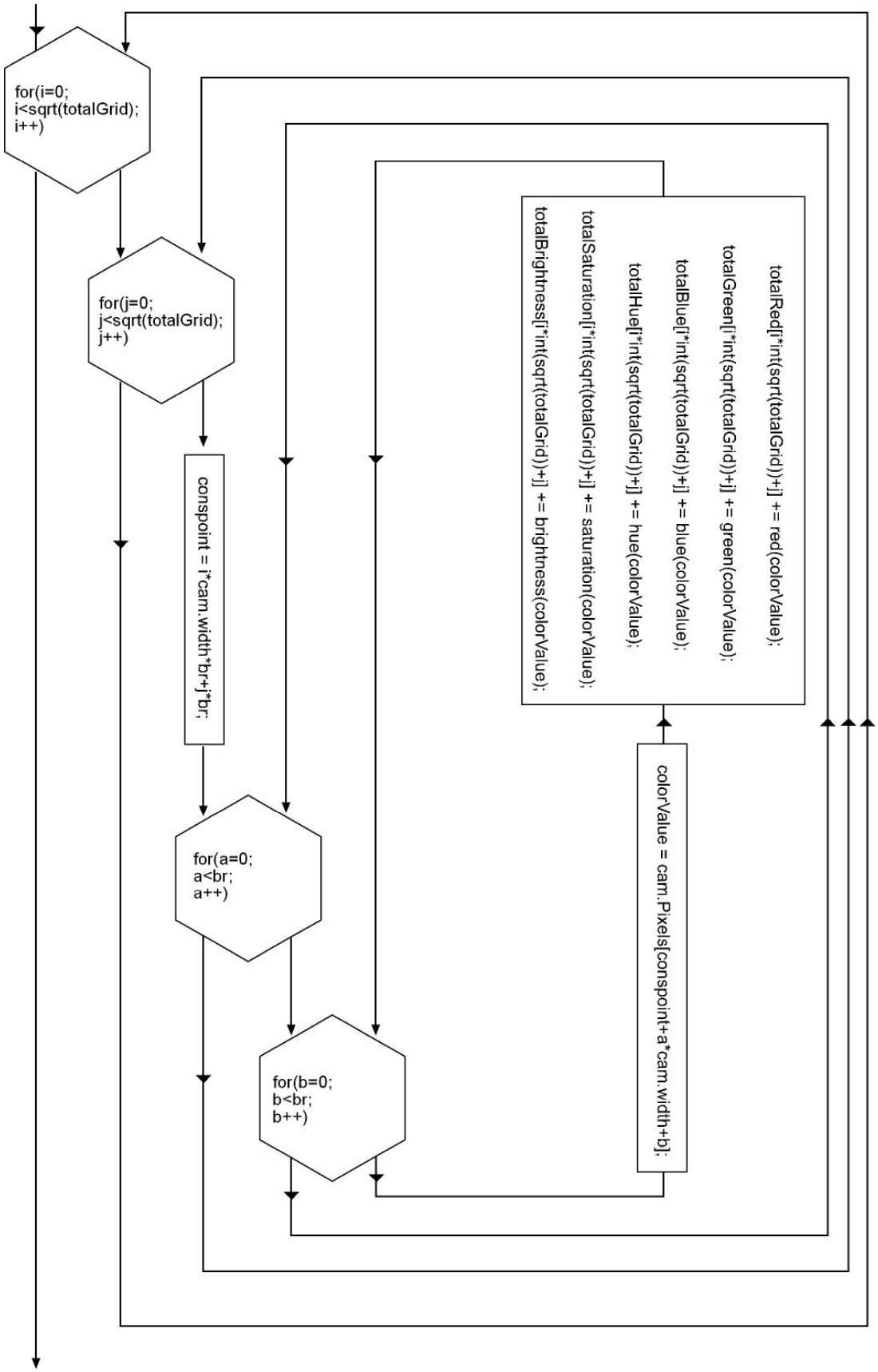
Üçüncü olarak, totalBlue dizisinin ilgili elemanına, blue( ) standart fonksiyonunun parametre girişine colorValue değişkeninin girilmesi ile elde edilen mavi değeri atanır.

Dördüncü olarak, totalHue dizisinin ilgili elemanına, hue( ) standart fonksiyonunun parametre girişine colorValue değişkeninin girilmesi ile elde edilen renk değeri atanır.

Beşinci olarak, totalSaturation dizisinin ilgili elemanına, saturation( ) standart fonksiyonunun parametre girişine colorValue değişkeninin girilmesi ile elde edilen doygunluk değeri atanır.

Altıncı olarak, totalBrightness dizisinin ilgili elemanına, brightness( ) standart fonksiyonunun parametre girişine colorValue değişkeninin girilmesi ile elde edilen parlaklık değeri atanır.

Burada kullandığımız, red( ), green( ), blue( ), hue( ), saturation( ) ve brightness( ) standart fonksiyonlarından dönen sonuçlar, 0 ile 255 değerleri arasında olmaktadır. Bütün bu değerler üst üste toplanarak totalRed, totalGreen, totalBlue, totalHue, totalSaturation, totalBrightness dizi değişkenlerinde saklanır. Böylece, her bir alt çerçevenin sahip olduğu piksellerin ilgili özelliklerinin toplam değerleri elde edilmiş olur. (Şekil 3.41)



Şekil 3.41 Her bir alt çerçevenin sahip olduğu piksellerin değerlerinin ilgili dizi değişkenlerinde toplanması

Böylece totalRed, totalGreen, totalBlue, totalHue, totalSaturation ve totalBrightness dizi değişkenlerinde, bütün alt çerçevelerin piksellerindeki toplam kırmızı, yeşil, mavi, renk, doygunluk ve parlaklık değerlerinin toplamı bulunmaktadır.

Piksel değerlerinin toplamlarının alınmasının ardından; totalRed, totalGreen, totalBlue, totalHue, totalSaturation ve totalBrightness dizi değişkenlerinin ilgili elemanlarının değerlerinin, her bir alt çerçevede piksel başına ne kadar değer düştüğü hesaplanır. Bunun için, bir alt çerçevedeki bu toplam değerler, bu alt çerçevenin toplam piksel sayısına bölünür. Böylece her alt çerçevede piksel başına düşen kırmızı, yeşil, mavi, renk, doygunluk ve parlaklık değerleri bulunur. Bu değerler 0 ile 255 değerleri arasında olurlar.

Bu işlem için, kullanıcının kontrolü ile belirlenen totalGrid değişkenini değeri kadar çalışacak bir döngü oluşturulur. Bu döngü içerisinde; bir alt çerçevedeki toplam değerler, bu alt çerçevenin toplam piksel sayısına bölünerek, piksel başına düşen değer hesaplanır.

Her alt çerçevenin toplam piksel sayısı, cam değişkeninin toplam piksel sayısının totalGrid değişkeninin değerine bölünmesi ile bulunur.

Döngü içerisinde ilk olarak; totalRed dizisinin döngü indeksi (i) ile belirlenen elemanının değeri bu alt çerçevenin toplam piksel sayısına bölünür. Ardından çıkan değer, averageRed dizisinin döngü indeksi (i) ile belirlenen elemanının değerine atanır. Böylece bu alt çerçevedeki piksellerin taşıdığı ortalama kırmızı değeri 0 ile 255 sayıları arasında belirlenmiş olur.

İkinci olarak; totalGreen dizisinin döngü indeksi (i) ile belirlenen elemanının değeri bu alt çerçevenin toplam piksel sayısına bölünür. Ardından çıkan değer, averageGreen dizisinin döngü indeksi (i) ile belirlenen elemanının değerine atanır. Böylece bu alt çerçevedeki piksellerin taşıdığı ortalama yeşil değeri 0 ile 255 sayıları arasında belirlenmiş olur.

Üçüncü olarak; totalBlue dizisinin döngü indeksi (i) ile belirlenen elemanının değeri bu alt çerçevenin toplam piksel sayısına bölünür. Ardından çıkan değer, averageBlue dizisinin döngü indeksi (i) ile belirlenen elemanının değerine atanır. Böylece bu alt çerçevedeki piksellerin taşıdığı ortalama mavi değeri 0 ile 255 sayıları arasında belirlenmiş olur.

Dördüncü olarak; totalHue dizisinin döngü indeksi (i) ile belirlenen elemanının değeri bu alt çerçevenin toplam piksel sayısına bölünür. Ardından çıkan değer, averageHue dizisinin döngü indeksi (i) ile belirlenen elemanının değerine atanır. Böylece bu alt çerçevedeki piksellerin taşıdığı ortalama renk değeri 0 ile 255 sayıları arasında belirlenmiş olur.

Beşinci olarak; totalSaturation dizisinin döngü indeksi (i) ile belirlenen elemanının değeri bu alt çerçevenin toplam piksel sayısına bölünür. Ardından çıkan değer, averageSaturation dizisinin döngü indeksi (i) ile belirlenen elemanının değerine atanır. Böylece bu alt çerçevedeki piksellerin taşıdığı ortalama doygunluk değeri 0 ile 255 sayıları arasında belirlenmiş olur.

Altıncı olarak; totalBrightness dizisinin döngü indeksi (i) ile belirlenen elemanının değeri bu alt çerçevenin toplam piksel sayısına bölünür. Ardından çıkan değer, averageBrightness dizisinin döngü indeksi (i) ile belirlenen elemanının değerine atanır. Böylece bu alt çerçevedeki piksellerin taşıdığı ortalama parlaklık değeri 0 ile 255 sayıları arasında belirlenmiş olur.

Bütün bu işlemlerden sonra görüntülerin işlenmesi ile kullanıcı için anlamlı parametrelere dönüştürülmesi işlemi sonlandırılmış olur. Bundan sonra, kullanıcı kendi kontrolleri ile bu değerler arasında istediği gibi ilişkiler yaratabilecek ve ses sinyal parametrelerini oluşturabilecektir.

3.3.4.3 Kullanıcı kontrolleri ile görüntü parametrelerinin birleştirilmesi ve synthStore dizisine aktarılması

ComputePixelValues( ) fonksiyonunda, görüntülerin işlenmesi ile parametrelerin oluşturulmasından sonra; sıra, oluşturulan parametrelerin kullanıcı kontrolleri ile birleştirilmesine gelir. Burada yapılması gereken averageRed, averageGreen, averageBlue, averageHue, averageSaturation ve averageBrightness dizilerinin değerlerinin, var olan her bir ses kanalı için kullanıcının belirlediği kontroller ile değerlendirilmesidir.

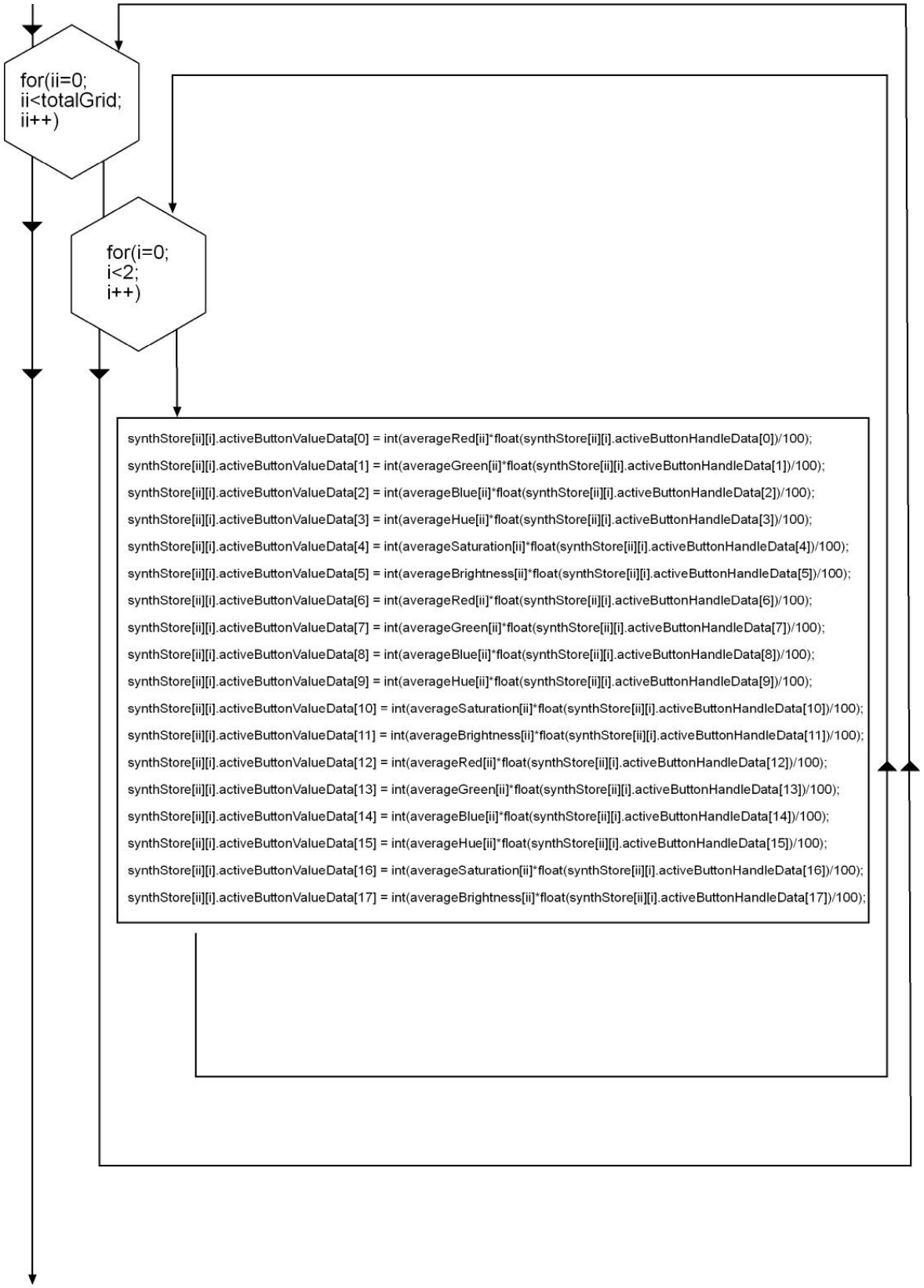
Görüntülerden gelen altı özellik (kırmızı, yeşil, mavi, renk, doygunluk, parlaklık) ses sinyallerinin üç parametresine (frekans, ses yüksekliği, pan) bağlanabildiğinden dolayı, toplam on sekiz adet birleştirilecek eleman vardır. Bu yüzden, ses sinyallerinin bütün parametrelerini tutan toplam otuz iki ses kanalı için otuz iki eleman barındıran synthStore dizisinin her bir elemanının activeButtonValueData özelliği on sekiz eleman boyutunda bir dizidir. Bu özellik, görüntüden alınan parametrelerin kullanıcı kontrolleri ile aldıkları son değerleri saklar. Burada bahsettiğimiz kullanıcı kontrolleri, işleme synthStore dizisinin activeButtonHandleData özelliği ile katılır. Bu özellik de, activeButtonValueData özelliği gibi on sekiz elemandan oluşur. Kullanıcının, her bir görüntü parametresinin toplam ses sinyaline katılım oranlarını belirlemek için kullandığı ve arayüzde bunlara denk düşen ActiveButtonHandle nesneleri ile belirlediği özelliklerdir. Bu özellik 0 ile 100 tamsayıları arasında bir değer alır.

Burada öncelikle, bütün ses sinyal kanallarının değerlerinin tamamlanabilmesi için synthStore dizisinin bütün elemanlarını kapsayacak iç içe iki for döngüsü tanımlanır. Bu döngülerden ilki, totalGrid değişkeninin taşıdığı değer kadar çalışır. İkinci döngü ise, her bir görüntü çerçevesine denk düşen iki ses sinyal kanalı için iki kere çalışır. Böylece synthStore dizisinin on altı sütun ve iki satırdan oluşan bütün elemanlarının ilgili değerleri işlenmiş olur.

Bu döngü içerisinde, synthStore dizisinin ilk döngü indeksi (ii) ve ikinci döngü indeksi (i) ile belirlenen elemanının activeButtonHandleData dizi özelliğinin ilgili elemanı, 100

ile bölünerek float tipinde 0 ile 1 değerleri arasında bir değer elde edilir. Ardından averageRed dizisinin ilk döngü indeksi (ii) ile belirlenen elemanının değeri ile çarpılır. Çıkan float değeri integer değerine çevrilir. Bu değer synthStore dizisinin ilk döngü indeksi (ii) ve ikinci döngü indeksi (i) ile belirlenen elemanının activeButtonValueData dizi özelliğinin ilgili elemanına atanır. Döngünün her çalışmasında, üç ses sinyal kanalı özelliği için (frekans, ses yüksekliği, pan) ve altı görüntü parametresi (kırmızı, yeşil, mavi, renk, doygunluk, parlaklık) için toplam on sekiz kere yukarıdaki işlemler gerçekleştirilir. Döngünün her çalışması sadece tek bir ses sinyal kanalının özelliklerini işler. Toplam otuz iki ses sinyal kanalı için synthStore dizisinin otuz iki elemanına bu işlemler uygulanır.

Böylece, artık otuz iki ses sinyal kanalının, görüntüye bağlı olan özelliklerinin görüntülerden alabileceği değerler kullanıcı kontrolleri ile belirlenmiş ve Synth sınıfı türündeki synthStore dizisinin activeButtonValueData dizi özelliği içerisinde saklanır hale getirilmiştir. (Şekil 3.42)



Şekil 3.42 Kullanıcı kontrolleri ile birleştirilmiş olan görüntü parametrelerinin synthStore dizisine aktarılması

Bu aşama sonlandığında, kullanıcının, görüntü parametrelerinin toplam ses sinyal oranlarını belirlediği ve arayüzde ActiveButtonHandle nesneleri ile gösterilen aşamaların hesaplamaları bitmiştir. Bundan sonra, kullanıcının kontrol imkanının olduğu bir sınamanın daha yapılması gerekmektedir. Bu sınama, görüntü parametrelerinin oranların belirlendiği ve ardından ses sinyaline parametre olarak katıldığı yolun açılıp kapanması kontrolüdür.

Burada yapılacak olan iş, öncelikle, alınan görüntü parametrelerine bağlanabilen üç adet ses sinyal parametresinin, el ile sabit değer vererek mi, yoksa görüntü parametrelerine bağlanarak mı belirleneceğinin kullanıcı tarafından nasıl seçildiğinin sınanmasıdır. Eğer kullanıcı görüntü değerlerinin belirleyici olmasını seçmiş ise, altı adet görüntü parametrelerinden hangilerinin bu ses sinyallerine katılacağı sınanacak ve bu parametreler için daha önce belirlenmiş olan oranlar birleştirilecektir. Ardından oluşan ortalamaların değerlerine göre, ses sinyal kanallarının parametreleri belirlenecektir. Bu belirlemenin yapılabilmesi için, görüntü değerlerinin kullanıcı kontrolleri ile birleşmesi sonucu oluşan yüzde değerinin, ses sinyalleri için gerekli olan özelliklerin spektrum aralıklarına yansıtılması gerekmektedir. Bu yansıtma işleminin ardından sythStore dizisinin ilgili elemanına bu değerler atanır.

Bu görevlerin yerine getirilmesi için öncelikle, üç adet iç içe for döngüsünün tanımlanması gerekmektedir. Birinci for döngüsü, totalGrid değişkeninin taşıdığı değer kadar çalışacaktır. İkinci for döngüsü, synthStore dizisinin satır sayısı kadar çalışacaktır. Üçüncü for döngüsü, synthStore dizisinin sahip olduğu monitoringManualData dizi özelliğinin eleman sayısı kadar çalışacaktır.

Bu iç içe döngünün içerisinde ilk olarak, synthStore dizisinin ilk döngü indeksi (iii) ve ikinci döngü indeksi (ii) ile belirlenen elemanının monitoringManualData dizi özelliğinin üçüncü döngü indeksi ile belirlenen elemanının ‘true’ ya da ‘false’ olup olmadığı sınanır. Bu sınama eğer ‘false’ ise hiçbir işlem yapılmaz ve bir sonraki döngüye geçilir. Eğer ‘true’ ise yukarıda anlatılan işlemlerin gerçekleştirilmesine başlanır.

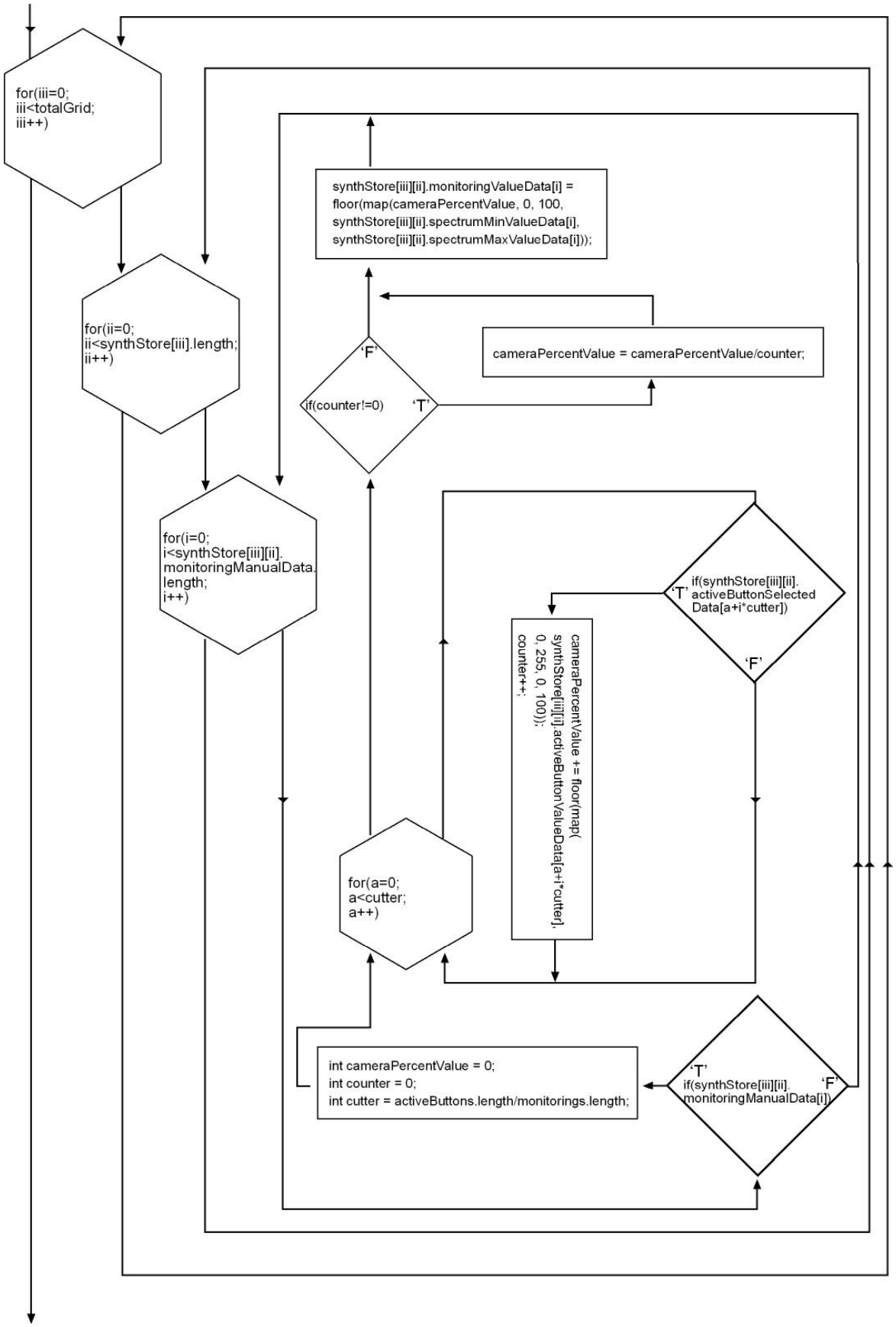
İlk olarak bu iç içe döngüler içerisinde gerekli olan ve etkinlik alanı sadece bu döngülerin içi olan yerel değişkenler tanımlanır. Bu değişkenlerden ilki, cameraPercentValue adında ve integer temel tipinde olan değişkendir. Bu değişken, kameradan gelen değerlerin kullanıcının kontrolleri ile tam olarak belirlenmesi ile oluşan son yüzde değerlerini belirler. Toplam altı adet görüntü parametresinden oluşan üç adet ses parametresinin değerlerinin üst ve alt spektrumlar arasında hangi yüzde değerinde olacağının bilgisini taşır. Tanımlanan ikinci değişken, kullanıcının kontrolleri ile belirlenen görüntü değerlerinden kaç tanesinin ortalamaya katıldığının bilgisini taşıyan bir değişkendir. Bu değişken counter adında ve integer temel tipinde tanımlanmış bir yerel değişkendir. Üçüncü değişken, monitorings dizisinde bulunan toplam üç adet elemana karşılık düşen activeButtons dizisi elemanlarının katsayı değerini tutar. Bu değişken, activeButtons dizisinin eleman sayısının monitorings dizisinin eleman sayısına bölünmesi ile bulunur. Böylece activeButtons dizisinden kaç elemanın, monitorings dizisinin tek elemanına denk düştüğünün bilgisi tutulur. Bu değişken cutter adında ve integer temel tipinde tanımlanmış bir yerel değişkendir. Bu yerel değişkenlerin hepsine ilk değer olarak 0 atanmıştır. Bu değişkenlerin for döngülerinin içinde tanımlanmasının nedeni, her eleman için bu değişkenlerin sıfırlanmasının sağlamasını yapmaı içindir.

Yerel değişkenlerin tanımlanmasının ardından, dördüncü bir for döngüsü tanımlanır. Bu döngü cutter değişkeninin değeri kadar çalışır. Bu döngü içerisinde ilk olarak, synthStore dizisinin birinci döngü (iii) ve ikinci döngü (ii) indeks değerine karşılık gelecek olan elemanının activeButtonSelectedData dizi özelliğinin, üçüncü döngü indeksi ile cutter değişkeninin çarpımı ve bununla dördüncü döngü indeksinin toplamına karşılık düşen elemanının değeri sınanır. Bu sınama, kullanıcının ActiveButton nesnesini aktif hale getirip getirmediğine göre ‘true’ ya da ‘false’ değeri alacaktır. Böylece sadece ilgili ActiveButton nesnelerinin seçili olanları toplamaya ve dolayısıyla ortalamaya katılacaktır. Bu sınama eğer ‘true’ çıkarsa; synthStore dizisinin birinci döngü (iii) ve ikinci döngü (ii) indeks değerine karşılık gelecek olan elemanının activeButtonValueData dizi özelliğinin, üçüncü döngü indeksi ile cutter değişkeninin çarpımı ve bununla dördüncü döngü indeksinin toplamına karşılık düşen elemanının 0 ile 255 arasında olan değeri, 0 ile 100 değerleri arasına yansıtılır. Bu çıkan değer,

cameraPercentValue değişkeninin üzerine eklenir. Ardından counter değişkeni, toplam kaç ActiveButton nesnesinin seçili olduğu bilgisini tutmak için 1 arttırılır.

Bu dördüncü döngünün çalışması sonlandırıldıktan, yani her bir monitorings dizisine denk düşen activeButtons dizisi elemanı sayısında çalıştıktan sonra, üçüncü döngü içerisinde bir sınama daha yapılır. Bu sınama, counter değişkeninin değerini sınar. Bu değişkenin değeri eğer 0 değilse, cameraPercentValue değişkeninin değeri counter sayısına bölünür ve yine kendisine atanır. Ardından cameraPercentValue değişkeninin 0 ile 100 arasında bulunan değeri synthStore dizisinin birinci döngü (iii) ve ikinci döngü (ii) indeks değerine karşılık gelecek olan elemanının sahip olduğu spectrumMinValueData ve spectrumMaxValueData dizi özelliklerinin üçüncü döngü indeksine karşılık gelen değerler arasına yansıtılır.

Bu işlemler sonucunda oluşan değer, synthStore dizisinin birinci döngü (iii) ve ikinci döngü (ii) indeks değerine karşılık gelecek olan elemanının monitoringValueData dizi özelliğinin üçüncü döngü indeksine karşılık gelen değerine atanır. (Şekil 3.43)



Şekil 3.43 Kullanıcı kontrolleriyle oluşan ortalamaların synthStore dizisine atanması

Böylece, var olan bütün Monitoring nesnelerine denk düşen ActiveButton ve ActiveButtonHandle nesnelerinin değerleri hesaplanmış ve ilgili synthStore dizisinin elemanlarına atanmıştır.

Kamera ile alınan görüntülerin işlenerek kırmızı, yeşil, mavi, renk, doygunluk ve parlaklık değerlerinin ölçülüp, ardından kullanıcı kontrolleri ile birleştirilip ses sinyal kanallarına gönderilecek olan parametrelerin yaratılması ve bu parametrelerin bütün ses kanallarının değerlerini tutan synthStore dizisine atanması işini üstlenen ComputePixelValues( ) fonksiyonu burada sonlanır. Yazılımın ana iskeletini oluşturan draw( ) döngü fonksiyonu içerisinde bulunan bu fonksiyonun sonlanmasının ardından, oluşturulan ses sinyal kanallarının parametreleri ile ses sinyallerinin oluşturulması işlemi başlar.

3.3.5 Ses Sinyal Parametrelerinden Ses Sinyallerinin Oluşturulması

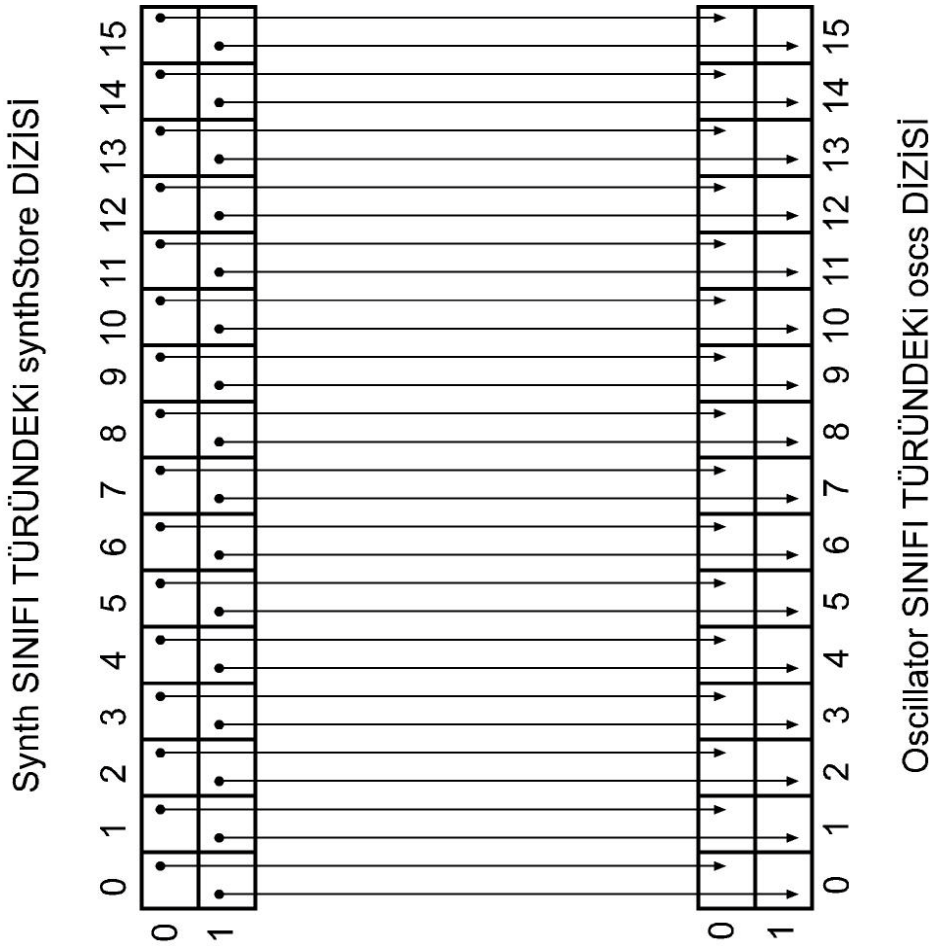
Yazılımın en temel amacı olan ses sinyallerini oluşturma işlemi, görüntü parametrelerinin ComputePixelValues( ) fonksiyonunda oluşturulmaları ve bunların synthStore dizisine aktarılması işlemlerinden sonra yapılır. Bunun gerçekleştirilmesi, draw( ) fonksiyonu içerisinde yer alan ve ComputePixelValues( ) fonksiyonundan sonra çalışmaya başlayan SoundGenerator( ) fonksiyonu içerisinde olur.

Asıl olarak burada gerçekleşen, ses sinyal çıkış kanallarının oluşturulması değildir. Çalışmanın ‘3.3.2.4.3’ no’lu kısmında anlatıldığı gibi ses sinyallerinin hoparlörlere gönderilmesi için gerekli olan kanallar AudioOutput sınıfından yazılımın setup( ) fonksiyonu içerisinde oluşturulur. Bunlara denk düşecek olan Oscillator sınıfı nesneleri de setup( ) fonksiyonunda yaratılır. Bunlar yazılımın çalıştığı her anda hazır olarak mevcutturlar. (Şekil 3.18)

SoundGenerator( ) fonksiyonunda gerçekleşen ise, kullanıcının kamerada yakaladığı görüntüler ve bu görüntülere bağlı olan ses sinyal parametreleri ile kullanıcı tarafından

belirlenen diğer ses sinyal özelliklerinin var olan kanallara aktarılması işlemidir. Böylece ses sinyalleri her seferinde tekrar güncellenerek oluşturulur.

Burada gerçekleştirilmesi gereken işlem, synthStore dizisinde bulunan elemanların hepsinin değerlerinin, Oscillator sınıfı nesneleri tutan oscs dizisinin ilgili elemanlarının özelliklerine aktarılmasıdır. (Şekil 3.44)



Şekil 3.44 oscs dizisindeki elemanların synthStore dizisindeki elemanlardan beslenmesi

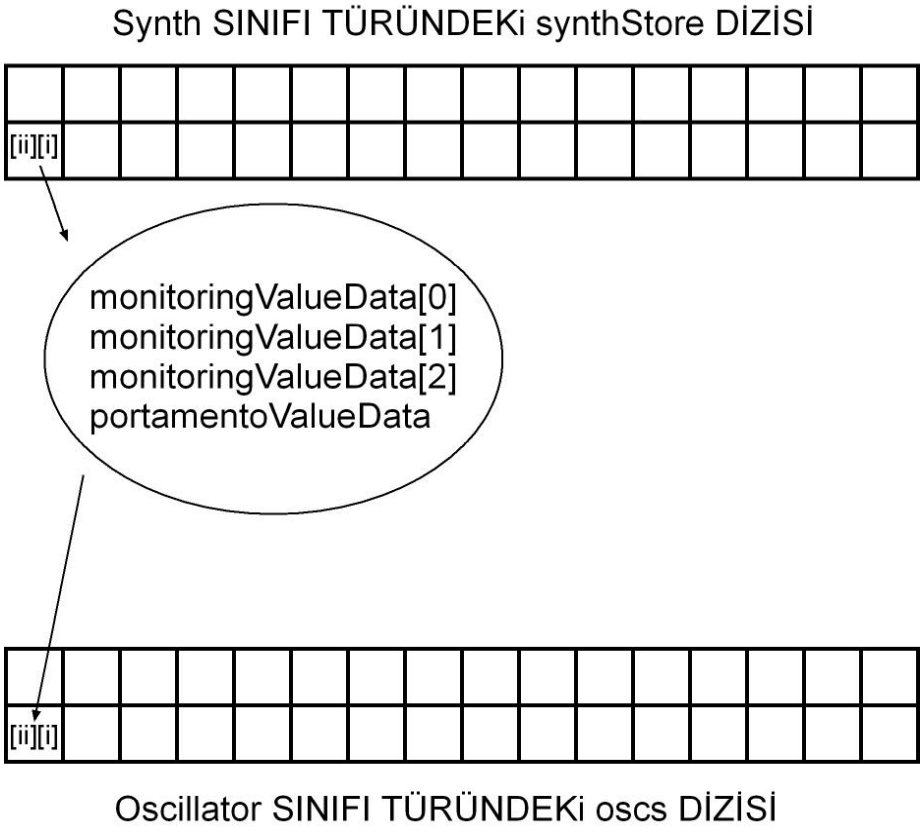
Bu işlemlerin gerçekleştirilmesi için, öncelikle iç içe iki for döngüsü oluşturulur. synthStore dizisinin bütün elemanlarına ulaşılabilmesi için, dolayısıyla oscs dizisi elemanlarına da ulaşılabilmesi için, birinci for döngüsü synthStore dizisinin sütun sayısı kadar ikinci for döngüsü de synthStore dizisinin satır sayısı kadar çalıştırılır.

Bu döngüler içerisinde ilk olarak; synthStore dizisinin birinci döngü indeksi (ii) ve ikinci döngü indeksi (i) ile işaret edilen elemanının sahip olduğu monitoringValueData dizi özelliğinin birinci elemanı (frekans değeri) oscs dizisinde karşılık olan elemanın (ii ve i indeksli eleman) frekans değerine atanır.

İkinci olarak; synthStore dizisinin birinci döngü indeksi (ii) ve ikinci döngü indeksi (i) ile işaret edilen elemanının sahip olduğu monitoringValueData dizi özelliğinin ikinci elemanı (ses yüksekliği değeri) oscs dizisinde karşılık olan elemanın (ii ve i indeksli eleman) ses yüksekliği değerine atanır. Bu işlem gerçekleştirilirken monitoringValueData dizisinin değeri 0 ile 100 arasında iken, 0 ile 1 arasına yansıtılır.

Üçüncü olarak; synthStore dizisinin birinci döngü indeksi (ii) ve ikinci döngü indeksi (i) ile işaret edilen elemanının sahip olduğu monitoringValueData dizi özelliğinin üçüncü elemanı (pan değeri) oscs dizisinde karşılık olan elemanın (ii ve i indeksli eleman) ses yüksekliği değerine atanır. Bu işlem gerçekleştirilirken monitoringValueData dizisinin değeri 0 ile 100 arasında iken, -1 ile 1 arasına yansıtılır.

Dördüncü olarak; synthStore dizisinin birinci döngü indeksi (ii) ve ikinci döngü indeksi (i) ile işaret edilen elemanının sahip olduğu portaButtonData özelliğinin değeri sınanır. Eğer, bu değer ‘true’ ise, oscs dizisinin birinci döngü indeksi (ii) ve ikinci döngü indeksi (i) ile işaret edilen elemanının portamento özelliğine giriş parametresi olarak synthStore dizisinde karşılık olan elemanın (ii ve i indeksli eleman) portamentoValueData değeri girilir. Eğer sınamada ‘false’ değeri çıkarsa, oscs dizisinin birinci döngü indeksi (ii) ve ikinci döngü indeksi (i) ile işaret edilen elemanının portamento özelliği kapatılır. (Şekil 3.45)



Şekil 3.45 synthStore dizisinden oscs dizisine değerleri aktarılan parametreler

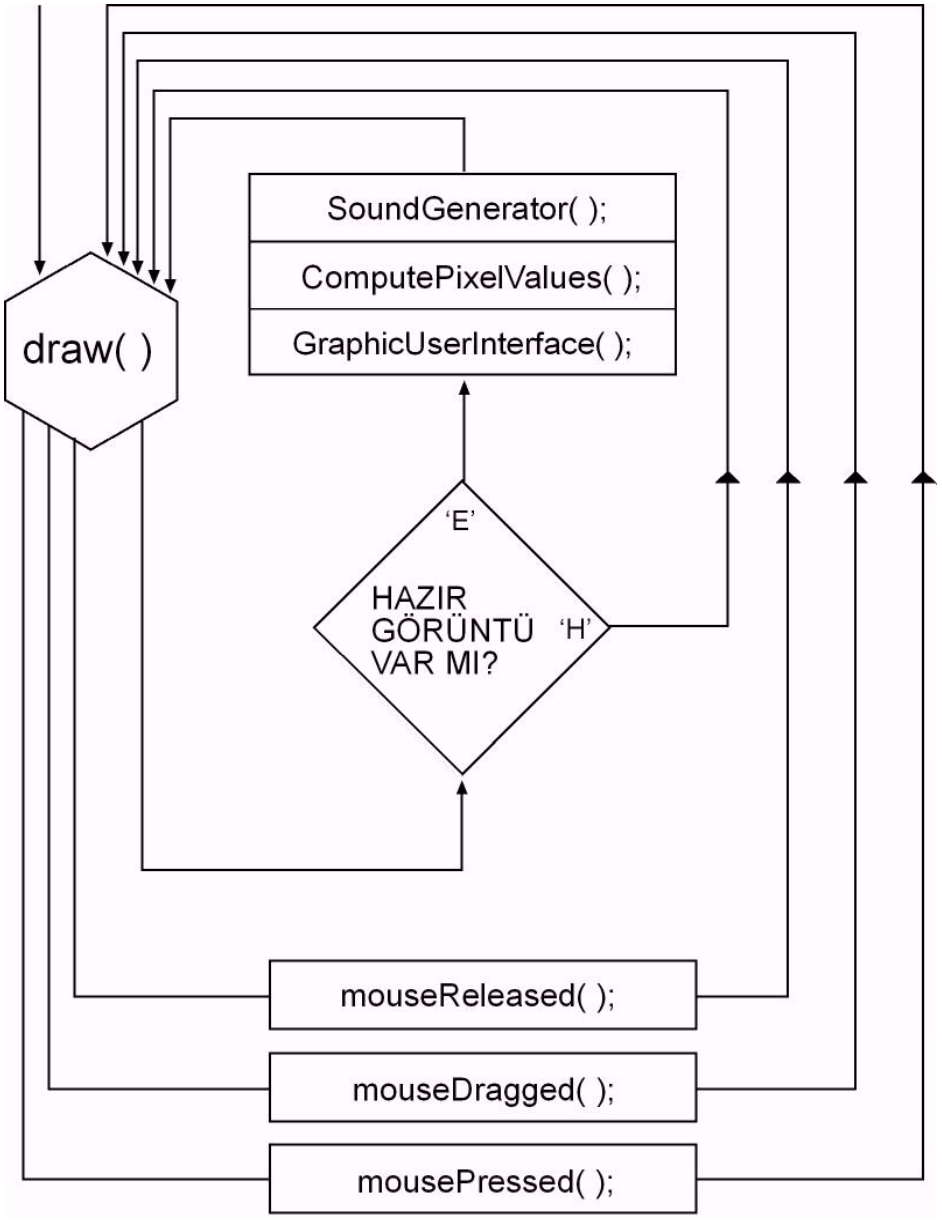
Burada görüldüğü gibi, ses sinyalleri için önemli olan dört özellik, SoundGenerator( ) fonksiyonunda oscs dizsinin elemanları olan Oscillator nesnelerine aktarılmıştır. Bu dört özellik dışında, ses sinyallerinin karakterlerini belirleyen bir diğer özellik olan ses sinyal türü bilgisi bu fonksiyon içerisinde oscs dizisi elemanlarına atanmamaktadır. Bunun nedeni, bu özelliğin değişmesi ile ses sürekliliğinde bir miktar kayıp olmasıdır. Çünkü, bu ses özelliğinin değişimi için ses sinyalinin yeniden yaratılmasına ihtiyaç vardır. Bu özellik değiştirildiğinde, oscs dizisinin ilgili elemanında tutulan ses sinyal nesnesi yeniden üretilir. Bu yüzden, ses dalgasının türü kullanıcı tarafından değiştirilmedikçe aynı kalacaktır. Sadece kullanıcının bu özelliği değiştirmek için ilgili arayüz nesneleri ile etkileşime geçtiğinde bu sinyaller yeniden üretilir ve oscs dizisine aktarılır. Bu sebeple, sürekli tekrarlanan SoundGenerator( ) fonksiyonunda bu özelliğin oscs dizisine aktarılmaması uygun görülmüştür.

Ses sinyal kanallarının parametre değerlerinin güncellenmesi işlemini gerçekleştiren SoundGenerator( ) fonksiyonunun sonlanmasıyla, yazılımın temel döngüsü olan draw( ) fonksiyonunun bir döngüde yaptığı işlemler tamamlanmıştır.

3.3.6 Kullanıcı Grafik Arayüzü İle Yapılan Mouse Etkileşimleri

Yazılımın çalışması, kullanıcının herhangi bir etkileşimi olmaksızın sürerse, lineer olarak devam eden bir yapı ile devam eder. Temel iskeleti oluşturan draw( ) fonksiyonu, sürekli kendisini tekrarlayarak kullanıcının uygulama çerçevesini kapatmasına kadar çalışmaya devam eder. Ancak kullanıcı sistemle herhangi bir etkileşime girerse, lineer olarak çalışan draw( ) fonksiyonunda bir kırılma gerçekleşir. Bu kırılmayı sağlayan, processing ortamının sürekli olarak yaptığı kontrol sınamalarıdır. Processing ortamının yaptığı sınamalara denk düşen standart fonksiyonlar bu kırılmaların işlevsel olarak değerlendirilmesini imkanlı kılar.

Bizim burada kullanarak kullanıcının sistem üzerinde kontroller sağlamasını imkanlı kılan ve sistemi non-lineer bir yapıya kavuşturan etkileşimler, mouse ile yapılan etkileşimler olacaktır. Bu etkileşimler; kullanıcının mouse butonuna basması, mouse butonunu sürüklemesi ve mouse butonuna basmayı sonlandırmasıdır. Bu etkileşimler gerçekleştiğinde draw( ) fonksiyonun döngüsünden çıkılır, ilgili durum sınamasının çağırdığı fonksiyonlar çalıştırılır ve ardından tekrar draw( ) fonksiyonunun döngüsüne geri dönülür. (Şekil 3.46)



Şekil 3.46 Sistem akışının mouse etkileşimleri ile draw( ) fonksiyonundan çıkması ve tekrar bu fonksiyona geri dönmesi

Yukarıda belirtilen etkileşimlere denk düşen ve processing ortamında standart olarak bulunan üç fonksiyon vardır. Bunlar, mousePressed( ), mouseDragged( ) ve mouseReleased( ) fonksiyonlarıdır. Kullanıcının mouse ile olan etkileşimlerinde kendiliğinden çağırılan bu fonksiyonlar, standart hallerinde herhangi bir giriş parametresi almazlar ve herhangi bir çıkış parametresi oluşturmazlar.

3.3.6.1 mousePressed( ) fonksiyonu

Processing ortamının standart fonksiyonlarından biridir. Bu fonksiyon, kullanıcı, aracın çalıştığı herhangi bir anda mouse butonuna bastığı zaman çağırılır. Bu fonksiyon çağrıldığı zaman, bu fonksiyonun içerisinde tanımlanmış ve mouse butonunun bu etkisi ile aktif olacak eylemler var ise bu eylemler gerçekleşir.

Bunun için, bu fonksiyon içerisinde mouse butonuna basılması ile ilgili bir etkileşime sahip olan bütün arayüz nesneleri için gerekli sınamalar ve işlemler gerçekleştirilir. Bu sınama ve işlemlerin yapıldığı arayüz nesneleri şunlardır:

- ActiveButtonHandle nesneleri: Bu nesneler ile arayüzde sürükleyerek etkileşime geçildiği için, öncelikle herhangi bir mouse butonuna basılma eyleminde bu nesnelerden birine mi basıldığı sınanır. Bu sınamanın yapılması için, ActiveButtonHandle nesnelerini tutan activeButtonHandles dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Sınama işlemi, bu nesnelerin sınıf fonksiyonlarından biri olarak yaratılan overRect( ) fonksiyonunun her activeButtonHandles dizisi elemanı için çalıştırılması ile yapılır. Eğer arayüzde bulunan herhangi bir ActiveButtonHandle nesnesinin üzerine mouse ile basıldı ise, bu sınama ‘true’ olarak döner. Sınaması ‘true’ olarak dönen ActiveButtonHandle nesnesinin selected özelliğine ‘true’ değeri atanır. Böylece herhangi bir sürükleme eyleminde kullanıcının sürüklemeye çalıştığı arayüz nesnesinin bu activeButtonHandles dizisi elemanı olduğu anlaşılacak ve bu nesnenin sürükleme fonksiyonu çalıştırılacaktır. Eğer bütün ActiveButtonHandle nesnelerinin overRect( ) fonksiyonu ‘false’ olarak dönerse, kullanıcının arayüzde bulunan herhangi bir ActiveButtonHandle nesnesinin üzerine mouse ile basmadığı anlaşılır.
- Monitoring nesneleri: Bu nesneler ile arayüzde sürükleyerek etkileşime geçildiği için, öncelikle herhangi bir mouse butonuna basılma eyleminde bu nesnelerden birine mi basıldığı sınanır. Bu sınamanın yapılması için, Monitoring nesnelerini tutan monitorings dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Sınama işlemi, bu nesnelerin sınıf fonksiyonlarından biri olarak yaratılan overRect( ) fonksiyonunun her Monitoring dizisi elemanı için çalıştırılması ile yapılır. Eğer arayüzde bulunan herhangi bir Monitoring

nesnesinin üzerine mouse ile basıldı ise, bu sınama ‘true’ olarak döner. Sınaması ‘true’ olarak dönen Monitoring nesnesinin selected özelliğine ‘true’ değeri atanır. Böylece herhangi bir sürükleme eyleminde kullanıcının sürüklemeye çalıştığı arayüz nesnesinin bu monitorings dizisi elemanı olduğu anlaşılacak ve bu nesnenin sürükleme fonksiyonu çalıştırılacaktır. Eğer bütün Monitoring nesnelerinin overRect( ) fonksiyonu ‘false’ olarak dönerse, kullanıcının arayüzde bulunan herhangi bir Monitoring nesnesinin üzerine mouse ile basmadığı anlaşılır.

- Spectrum nesneleri: Bu nesneler ile arayüzde sürükleyerek etkileşime geçildiği için, öncelikle herhangi bir mouse butonuna basılma eyleminde bu nesnelerden birine mi basıldığı sınanır. Bu sınamanın yapılması için, Spectrum nesnelerini tutan spectrums dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Sınama işlemi, bu nesnelerin sınıf fonksiyonlarından biri olarak yaratılan overRect( ) fonksiyonunun her spectrums dizisi elemanı için çalıştırılması ile yapılır. Burada diğer arayüz nesnelerinden farklı olarak, her bir Spectrum nesnesinin üst sınır ve alt sınır olmak üzere iki adet kulpa sahip olmasından dolayı, bu sınamalar her Spectrum nesnesi için iki defa yapılır. Eğer arayüzde bulunan herhangi bir Spectrum nesnesinin üzerine mouse ile basıldı ise, bu sınama ‘true’ olarak döner. Sınaması ‘true’ olarak dönen Spectrum nesnesi kulpu eğer alt sınırı gösteren kulp ise minSelected özelliğine ‘true’ değeri atanır. Sınaması ‘true’ olarak dönen Spectrum nesnesi kulpu eğer üst sınırı gösteren kulp ise maxSelected özelliğine ‘true’ değeri atanır. Böylece herhangi bir sürükleme eyleminde kullanıcının sürüklemeye çalıştığı arayüz nesnesinin bu spectrums dizisi elemanı olduğu anlaşılacak ve bu nesnenin sürükleme fonksiyonu çalıştırılacaktır. Bu durumda iki adet bulunan kulplardan hangisinin özelliği ‘true’ değeri aldı ise diğerine ‘false’ değeri atanır. Eğer bütün Spectrum nesnelerinin overRect( ) fonksiyonu ‘false’ olarak dönerse, kullanıcının arayüzde bulunan herhangi bir Spectrum nesnesinin üzerine mouse ile basmadığı anlaşılır.
- PortamentoTool nesnesi: Bu nesne ile arayüzde sürükleyerek etkileşime geçildiği için, öncelikle herhangi bir mouse butonuna basılma eyleminde bu nesneye mi basıldığı sınanır. Sınama işlemi, bu nesnenin sınıf fonksiyonlarından

biri olarak yaratılan overRect( ) fonksiyonunun nesne için çalıştırılması ile yapılır. Eğer arayüzde bulunan PortamentoTool nesnesinin üzerine mouse ile basıldı ise, bu sınama 'true' olarak döner. Sınaması 'true' olarak dönen PortamentoTool nesnesinin selected özelliğine 'true' değeri atanır. Böylece herhangi bir sürükleme eyleminde kullanıcının sürüklemeye çalıştığı arayüz nesnesinin bu PortamentoTool nesnesi olduğu anlaşılacak ve bu nesnenin sürükleme fonksiyonu çalıştırılacaktır. Eğer PortamentoTool nesnesinin overRect( ) fonksiyonu 'false' olarak dönerse, kullanıcının arayüzde bulunan PortamentoTool nesnesinin üzerine mouse ile basmadığı anlaşılır.

3.3.6.2 mouseDragged( ) fonksiyonu

Processing ortamının standart fonksiyonlarından biridir. Bu fonksiyon, kullanıcı, aracın çalıştığı herhangi bir anda mouse butonuna basılı tutarak sürükleme işlemini gerçekleştirdiği zaman çağırılır. Bu fonksiyon çağrıldığı zaman, bu fonksiyonun içerisinde tanımlanmış ve mouse butonunun bu etkisi ile aktif olacak eylemler var ise bu eylemler gerçekleşir.

Bunun için, bu fonksiyon içerisinde mouse butonuna basılması ile ilgili bir etkileşime sahip olan bütün arayüz nesneleri için gerekli sınamalar ve işlemler gerçekleştirilir. Bu sınama ve işlemlerin yapıldığı arayüz nesneleri şunlardır:

- ActiveButtonHandle nesneleri: Bu nesneler ile arayüzde sürükleyerek etkileşime geçildiği için, herhangi bir mouse sürükleme eyleminde bu nesnelerden birinin mi sürüklendiği sınanır. Bu sınamanın yapılması için, ActiveButtonHandle nesnelerini tutan activeButtonHandles dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Bu döngü içerisinde, activeButtonHandles dizisinin elemanlarının selected özelliği sınanır. Eğer herhangi bir ActiveButtonHandle nesnesinin selected özelliği 'true' ise, kullanıcının bu nesneyi sürüklediği anlaşılır. Ardından, bu nesnenin sınıf yöntemi olarak tanımlanan mouseDragged( ) fonksiyonu çalıştırılır. ActiveButtonHandle sınıfından üretilen nesnelerde bulunan mouseDragged( ) fonksiyonunda, öncelikle nesnenin selected özelliği sınanır. Eğer bu özellik 'true' ise, nesnenin arayüzdeki yatay lokasyonu, mouse işaretinin yatay lokasyonuna eşitlenir. Bu eşitleme yapılırken,

nesnenin yatay özelliğinin değerinin, minimum noktaları ve maksimum noktaları arasında kalması için gerekli sınamalar yapılır. Ardından kulpun aldığı yeni yatay lokasyonuna göre hangi değeri aldığı hesaplanır. Bu hesaplamanın ardından yeni değer integer temel tipinde geri dönülür. Bu işlemler gerçekleştirildikten sonra, arayüzde bulunan bu ActiveButtonHandle nesnesinin değeri, synthStore dizisinin güncel elemanının, yani o sırada arayüzde görülmekte olan elemanının, activeButtonHandleData dizi özelliğinin ilgili elemanına atanır. Eğer herhangi bir ActiveButtonHandle nesnesinin selected özelliği 'true' değeri taşımıyorsa, kullanıcının bu nesneleri sürüklemediği anlaşılır.

- Spectrum nesneleri: Bu nesneler ile arayüzde sürükleyerek etkileşime geçildiği için, herhangi bir mouse sürükleme eyleminde bu nesnelerden birinin mi sürüklendiği sınanır. Bu sınamanın yapılması için, Spectrum nesnelerini tutan spectrums dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Bu döngü içerisinde, spectrums dizisinin elemanlarının minSelected ve maxSelected özellikleri sınanır. Eğer herhangi bir Spectrum nesnesinin minSelected ve maxSelected özellikleri 'true' ise, kullanıcının bu nesneyi sürüklediği anlaşılır. Ardından, bu nesnenin sınıf yöntemi olarak tanımlanan mouseDragged( ) fonksiyonu çalıştırılır. Spectrum sınıfından üretilen nesnelerde bulunan mouseDragged( ) fonksiyonunda, öncelikle nesnenin minSelected ve maxSelected özelliği sınanır. Eğer bu özelliklerden minSelected 'true' ise, nesnenin alt sınır belirleyen kulpun arayüzdeki dikey lokasyonu, mouse işaretinin dikey lokasyonuna eşitlenir. Bu eşitleme yapılırken, nesnenin dikey özelliğinin değerinin, minimum noktası ile üst sınırı gösteren kulpun dikey lokasyon değeri arasında kalması için gerekli sınamalar yapılır. Ardından kulpun aldığı yeni dikey lokasyona göre hangi değeri aldığı hesaplanır. Bu hesaplamanın ardından yeni değer minSpectrum değişkenine integer temel tipinde atanarak geri dönülür. Eğer bu özelliklerden maxSelected 'true' ise, nesnenin üst sınır belirleyen kulpun arayüzdeki dikey lokasyonu, mouse işaretinin dikey lokasyonuna eşitlenir. Bu eşitleme yapılırken, nesnenin dikey özelliğinin değerinin, maksimum noktası ile alt sınırı gösteren kulpun dikey lokasyon değeri arasında kalması için gerekli sınamalar yapılır. Ardından kulpun

aldığı yeni dikey lokasyona göre hangi değeri aldığı hesaplanır. Bu hesaplamanın ardından yeni değer maxSpectrum değişkenine atanarak integer temel tipinde geri dönülür. Bu işlemler gerçekleştirildikten sonra, arayüzde bulunan bu Spetrum nesnesinin sahip olduğu kulpların değerleri, synthStore dizisinin güncel elemanının, yani o sırada arayüzde görülmekte olan elemanının, spectrumMinValueData ve spectrumMaxValueData dizi özelliklerinin ilgili elemanına atanır. Eğer herhangi bir Spectrum nesnesinin minSelected ya da maxSelected özelliği 'true' değeri taşımıyorsa, kullanıcının bu nesneleri sürüklemediği anlaşılır.

- Monitoring nesneleri: Bu nesneler ile arayüzde sürükleyerek etkileşime geçildiği için, herhangi bir mouse sürükleme eyleminde bu nesnelerden birinin mi sürüklendiği sınanır. Bu sınamanın yapılması için, Monitoring nesnelerini tutan monitorings dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Bu döngü içerisinde, monitorings dizisinin elemanlarının selected özelliği ve monitoringManuals dizisinin ilgili indeksin işaret ettiği elemanının value özelliği sınanır. Eğer herhangi bir Monitoring nesnesinin selected özelliği ve monitoringManuals dizisinin value özelliği 'true' ise, kullanıcının bu Monitoring nesnesini sürüklediği anlaşılır. Burada her iki özelliğin de sınanmasının nedeni, kullanıcının Monitoring nesnesinin kulpunu sürükleyebilmesi için, kamera kaynağının kapatılması gerekliliğidir. Eğer kameradan gelen değerler kapalı değil ise, bu kaynaktan gelen değerler belirleyici olacağı için, kullanıcı Monitoring nesnesinin kulpunu sürükleyemez. Ardından, bu nesnenin sınıf yöntemi olarak tanımlanan mouseDragged( ) fonksiyonu çalıştırılır. Ancak bu fonksiyonun iki giriş parametresi vardır. Bunlar, ilgili Monitoring nesnesine karşılık gelen Spectrum nesnesinin alt sınır ve üst sınır belirleyen kulplarının dikey konumlarıdır. Monitoring sınıfından üretilen nesnelerde bulunan mouseDragged( ) fonksiyonunda, öncelikle nesnenin selected özelliği sınanır. Eğer bu özellik 'true' ise, nesnenin arayüzdeki dikey lokasyonu, mouse işaretinin dikey lokasyonuna eşitlenir. Bu eşitleme yapılırken, nesnenin dikey özelliğinin değerinin, giriş parametresi olarak alınan alt sınır spektrum kulpu ve üst sınır spektrum kulpunun dikey noktaları arasında kalması için gerekli sınamalar yapılır. Sınamaların ardından kulpun aldığı yeni dikey lokasyonuna

göre hangi değeri aldığı hesaplanır. Bu hesaplamanın ardından yeni değer integer temel tipinde geri dönülür. Bu işlemler gerçekleştirildikten sonra, arayüzde bulunan bu Monitoring nesnesinin değeri, synthStore dizisinin güncel elemanının, yani o sırada arayüzde görülmekte olan elemanının, monitoringValueData dizi özelliğinin ilgili elemanına atanır. Eğer herhangi bir Monitoring nesnesinin selected özelliği 'true' değeri taşımıyorsa, kullanıcının bu nesneleri sürüklemediği anlaşılır.

- PortamentoTool nesnesi: Bu nesne ile arayüzde sürükleyerek etkileşime geçildiği için, herhangi bir mouse sürükleme eyleminde bu nesnenin mi sürüklendiği sınanır. Bu sınamanın yapılması için, PortamentoTool nesnesini tutan portamento değişkeninin selected özelliği ve portabutton değişkeninin selected özelliği sınanır. Eğer PortamentoTool nesnesinin selected özelliği ve portabutton değişkeninin selected özelliği 'true' ise, kullanıcının bu PortamentoTool nesnesini sürüklediği anlaşılır. Burada her iki özelliğin de sınanmasının nedeni, kullanıcının PortamentoTool nesnesinin kulpunu sürükleyebilmesi için, portabutton değişkeninin açılması gerekliliğidir. Eğer portabutton değişkeni kapalı ise, ses kanalı için portamento özelliği kapalı olacağı için, kullanıcı PortamentoTool nesnesinin kulpunu sürükleyemez. Sınamaların ardından, bu nesnenin sınıf yöntemi olarak tanımlanan mouseDragged( ) fonksiyonu çalıştırılır. PortamentoTool sınıfından üretilen nesnelerde bulunan mouseDragged( ) fonksiyonunda, öncelikle nesnenin selected özelliği sınanır. Eğer bu özellik 'true' ise, nesnenin arayüzdeki yatay lokasyonu, mouse işaretinin yatay lokasyonuna eşitlenir. Bu eşitleme yapılırken, nesnenin yatay özelliğinin değerinin, minimum ve maksimum değerlerini gösteren yatay noktalar arasında kalması için gerekli sınamalar yapılır. Ardından kulpun aldığı yeni yatay lokasyonuna göre hangi değeri aldığı hesaplanır. Bu hesaplamanın ardından yeni değer integer temel tipinde geri dönülür. Bu işlemler gerçekleştirildikten sonra, arayüzde bulunan bu PortamentoTool nesnesinin değeri, synthStore dizisinin güncel elemanının, yani o sırada arayüzde görülmekte olan elemanının portamentoValueData özelliğine atanır. Eğer portamentoValueData nesnesinin selected özelliği 'true' değeri taşımıyorsa, kullanıcının bu nesneyi sürüklemediği anlaşılır.

3.3.6.3 mouseReleased( ) fonksiyonu

Processing ortamının standart fonksiyonlarından biridir. Bu fonksiyon, kullanıcı, aracın çalıştığı herhangi bir anda mouse butonunun basılı tutulmasına son verme işlemini gerçekleştirdiği zaman çağırılır. Bu fonksiyon çağrıldığı zaman, bu fonksiyonun içerisinde tanımlanmış ve mouse butonunun bu etkisi ile aktif olacak eylemler var ise bu eylemler gerçekleşir.

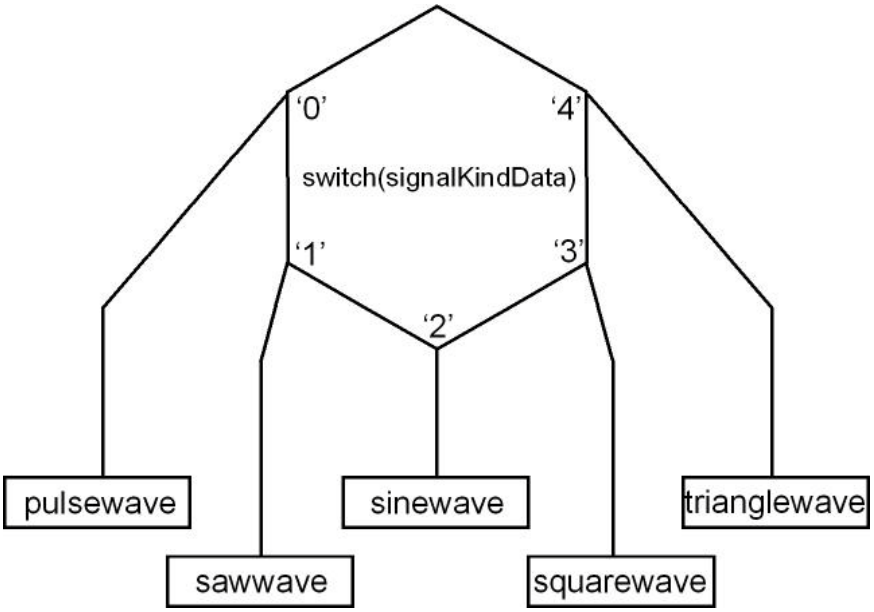
Bunun için, bu fonksiyon içerisinde mouse butonuna basılmasına son verilmesi ile ilgili bir etkileşime sahip olan bütün arayüz nesneleri için gerekli sınamalar ve işlemler gerçekleştirilir. Bu sınama ve işlemlerin yapıldığı arayüz nesneleri şunlardır:

- ActiveButton nesneleri: Bu nesneler ile arayüzde mouse butonuna basılması ve kaldırılması ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, bu eylemin bu nesnelerden birinin üzerinde mi yapıldığı sınanır. Bu sınamanın yapılması için, ActiveButton nesnelerini tutan activeButtons dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Burada öncelikle mouse butonunun basılı olma durumundan çıktığı anda, bu nesnelerden birinin üzerinde olup olmadığı, ActiveButton sınıfı nesnelerinin sahip olduğu overRect( ) fonksiyonu ile sınanır. Bu sınama 'true' değerini döner ise, ilgili ActiveButton nesnesinin mouseReleased( ) yöntemi çalıştırılır. Bu yöntem içerisinde, öncelikle bu nesnenin selected özelliği sınanır. Bu özellik 'true' ise, bu değişkene 'false' değeri atanır. Eğer bu özellik 'false' ise, bu değişkene 'true' değeri atanır. Bu işlemler gerçekleştirildikten sonra, arayüzde bulunan bu ActiveButton nesnesinin selected özelliğinin değeri, synthStore dizisinin güncel elemanının, yani o sırada arayüzde görülmekte olan elemanının, activeButtonSelectedData özelliğine atanır. Eğer ActiveButton sınıfı nesnelerinin sahip olduğu overRect( ) fonksiyonu ile yapılan sınamadan 'false' değeri döndüyse, kullanıcının mouse butonundan parmağını çekdiği sırada, mouse işaretçisinin bu nesne üzerinde olmadığı anlaşılır.
- ActiveButtonHandle nesneleri: Bu nesneler ile arayüzde sürükleme ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, mouse butonu basılı değilken herhangi bir sürükleme yapılamayacağı için bütün ActiveButtonHandle nesnelerinin selected özelliği 'false' yapılır. Bu işlemin

gerçekleşmesi için, ActiveButtonHandle nesnelerini tutan activeButtonHandles dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır.

- Spectrum nesneleri: Bu nesneler ile arayüzde sürükleme ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, mouse butonu basılı değilken herhangi bir sürükleme yapılamayacağı için bütün Spectrum nesnelerinin minSelected ve maxSelected özelliği 'false' yapılır. Bu işlemin gerçekleşmesi için, Spectrum nesnelerini tutan spectrums dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır.
- Monitoring nesnesi: Bu nesneler ile arayüzde sürükleme ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, mouse butonu basılı değilken herhangi bir sürükleme yapılamayacağı için bütün Monitoring nesnelerinin selected özelliği 'false' yapılır. Bu işlemin gerçekleşmesi için, Monitoring nesnelerini tutan monitorings dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır.
- MonitoringManual nesneleri: Bu nesneler ile arayüzde mouse butonuna basılması ve kaldırılması ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, bu eylemin bu nesnelerden birinin üzerinde mi yapıldığı sınanır. Bu sınamanın yapılması için, MonitoringManual nesnelerini tutan monitoringManuals dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Burada öncelikle mouse butonunun basılı olma durumundan çıktığı anda, bu nesnelerden birinin üzerinde olup olmadığı, MonitoringManual sınıfı nesnelerinin sahip olduğu overRect( ) fonksiyonu ile sınanır. Bu sınama 'true' değerini döner ise, ilgili MonitoringManual nesnesinin mouseReleased( ) yöntemi çalıştırılır. Bu yöntem içerisinde, öncelikle bu nesnenin value özelliği sınanır. Bu özellik 'true' ise, bu özelliğe 'false' değeri atanır. Eğer bu özellik 'false' ise, bu özelliğe 'true' değeri atanır. Bu işlemler gerçekleştirildikten sonra, arayüzde bulunan bu MonitoringManual nesnesinin value özelliğinin değeri, synthStore dizisinin güncel elemanının, yani o sırada arayüzde görülmekte olan elemanının, monitoringManualData özelliğine atanır. Eğer MonitoringManual sınıfı nesnelerinin sahip olduğu overRect( ) fonksiyonu ile yapılan sınamadan 'false' değeri döndüyse, kullanıcının mouse butonundan parmağını çekdiği sırada, mouse işaretçisinin bu nesne üzerinde olmadığı anlaşılır.

- signalKindSelections dizisi elemanları: Bu nesneler ile arayüzde mouse butonuna ile basılması ve kaldırılması ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, bu eylemin bu nesnelerden birinin üzerinde mi yapıldığı sınanır. Bu sınamanın yapılması için, tutan signalKindSelections dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Burada öncelikle mouse butonunun basılı olma durumundan çıktığı anda, bu nesnelerden birinin üzerinde olup olmadığı, signalKindSelections dizisi elemanlarının sahip olduğu overRect( ) fonksiyonu ile sınanır. Bu sınama 'true' değerini döner ise, mouse işleci arayüz üzerinde signalKindSelections dizisindeki diğer elemanların da üstünde olamayacağına göre, bütün signalKindSelections dizisi elemanlarının selected özelliği 'false' yapılır. Ardından, mouse butonundan parmak çekildiği anda, mouse işaretçisinin üstünde bulunduğu signalKindSelections dizisindeki elemanın mouseReleased( ) yöntemi çalıştırılır. Bu yöntem içerisinde, bu nesnenin selected özelliği sınanır. Bu özellik 'true' ise 'false' değeri atanır. Eğer 'false' ise 'true' değeri atanır. Bir önceki adımda, bütün signalKindSelections dizisi elemanlarının selected özelliğine 'false' değeri atandığı için, sadece bu nesneye 'true' değeri atanmış olur. Bu işlemler gerçekleştirildikten sonra, arayüzde bulunan bu nesnesinin signalKindSelections dizisi içerisinde taşıdığı indeks değeri, synthStore dizisinin güncel elemanının, yani o sırada arayüzde görülmekte olan elemanının, signalKindData özelliğine atanır. Ardından bu seçime göre, ses sinyal dalgasının türünde bir değişiklik olduğuna göre, outs dizisinin güncel olan elemanının içersinde bulunan ses sinyallerinde değişiklik yapılmak üzere, bu kanal eski ses sinyallerinden temizlenir. Sonrasında, kullanıcının seçimine göre belirlenen güncel synthStore dizisi elemanının signalKindData özelliğine göre belirlenen türde ses sinyali üretilerek bu elemana karşılık gelen oscs dizisi elemanına atanır. Burada, kullanıcının beş farklı ses sinyal türü üretme imkanı vardır. Bunlar pulswave, sawwave, sinewave, squarewave, trianglewave sinyal türleridir. Bu sinyaller üretildiğinde, 0 frekans değerinde, 0 ses yüksekliği değerinde ve çıkış kanallarındaki samplerate değerine eşit olarak 512 samplerate değerinde olurlar. (Şekil 3.47)



Şekil 3.47 Kullanıcının beş adet SignalKindSelection dizisi elemanı ile seçebildiği ses sinyal dalgası türleri

Üretilen ses sinyali, oscs dizisinin güncel elemanına atanır. Ardından oscs dizisinin bu yeni elemanı outs dizisinde kendisine karşılık gelen çıkış kanalına yüklenir. Yeni ses sinyal nesneleri üretilirken, kullanıcı sadece ses sinyal dalgası türünü değiştirmiştir. Ancak yukarıda gösterildiği gibi, yeni ses sinyali 0 frekans ve 0 ses yüksekliği değerine sahiptir. Burada kullanıcının kontrollerinde herhangi bir aksama olmaması için eski frekans, ses yüksekliği ve pan değerleri yeni üretilen ses sinyaline yüklenir.

- signalNumberSelections dizisi elemanları: Bu nesneler ile arayüzde mouse butonuna ile basılması ve kaldırılması ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, bu eylemin bu nesnelerden birinin üzerinde mi yapıldığı sınanır. Bu sınamanın yapılması için, signalKindSelections dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Burada öncelikle mouse butonunun basılı olma durumundan çıktığı anda, bu nesnelerden birinin üzerinde olup olmadığı, signalKindSelections dizisinin elemanlarının sahip olduğu overRect( ) fonksiyonu ile sınanır. Bu sınama 'true' değerini döner ise, mouse işleci arayüz üzerinde signalKindSelections dizisindeki diğer elemanların da üstünde olamayacağına göre, bütün signalKindSelections dizisi elemanlarının selected özelliği 'false' yapılır. Ardından, mouse butonundan parmağın çekildiği anda, mouse

işaretçisinin üzerinde bulunduğu signalKindSelections dizisindeki elemanın mouseReleased( ) yöntemi çalıştırılır. Bu yöntem içerisinde, bu nesnenin selected özelliği sınanır. Bu özellik 'true' ise 'false' değeri atanır. Eğer 'false' ise 'true' değeri atanır. Bir önceki adımda, bütün signalKindSelections dizisi elemanlarının selected özelliğine 'false' değeri atandığı için, sadece bu nesneye 'true' değeri atanmış olur. Bundan sonra, signalKindSelections dizisinin selected özelliği 'true' olan elemanının indeks değeri, synthStore dizisinde güncel satır indeksini belirlemek üzere global cCol değişkenine atanır.

- gridNumberSelections dizisi elemanları: Bu nesneler ile arayüzde mouse butonuna ile basılması ve kaldırılması ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, bu eylemin bu nesnelerden birinin üzerinde mi yapıldığı sınanır. Bu sınamanın yapılması için, gridNumberSelections dizisinin eleman sayısı kadar çalışan bir for döngüsü tanımlanır. Burada öncelikle mouse butonunun basılı olma durumundan çıktığı anda, bu nesnelerden birinin üzerinde olup olmadığı, gridNumberSelections dizisinin elemanlarının sahip olduğu overRect( ) fonksiyonu ile sınanır. Bu sınama 'true' değerini döner ise, mouse işleci arayüz üzerinde gridNumberSelections dizisindeki diğer elemanların da üstünde olamayacağına göre, bütün gridNumberSelections dizisi elemanlarının selected özelliği 'false' yapılır. Ardından, mouse butonundan parmağın çekildiği anda, mouse işaretçisinin üzerinde bulunduğu gridNumberSelections dizisindeki elemanın mouseReleased( ) yöntemi çalıştırılır. Bu yöntem içerisinde, bu nesnenin selected özelliği sınanır. Bu özellik 'true' ise 'false' değeri atanır. Eğer 'false' ise 'true' değeri atanır. Bir önceki adımda, bütün gridNumberSelections dizisi elemanlarının selected özelliğine 'false' değeri atandığı için, sadece bu nesneye 'true' değeri atanmış olur. Bundan sonra, gridNumberSelections dizisinin selected özelliği 'true' olan elemanının indeks değeri, kullanıcının belirlemiş olduğu alt çerçeve sayısının bilgisini tutmak üzere global gridNumber değişkenine atanır.

Kullanıcının grid sayısını değiştirmesinin ardından, sistemde yapılması gereken ayarlamalar vardır. Bu ayarlamalarda yapılacak olan, yeni görüntü alt çerçeve

sayısına bağlı olarak oluşan ses kanallarına ilk değerlerinin atanmasıdır. Aracın ilk çalışmaya başladığı anda hazır olan ses kanalları değerleri, burada da aynen kullanılır. Bu ilk değerlerin ses kanallarına aktarılması, bu değerlerin synthStore dizisi elemanlarına atanması ile gerçekleşir. Bunun için öncelikle iç içe iki for döngüsü kurulur. Birinci for döngüsü synthStore dizisinin yatay eleman sayısı kadar, ikinci for döngüsü synthStore dizisinin dikey eleman sayısı kadar çalışır. Ardından sırasıyla, synthStore dizisi elemanlarına ilk değerlerin atanması işlemlerine başlanır. İlk olarak, synthStore dizisinin activeButtonValueData dizi özelliğinin elemanları için, activeButtonValueData dizisinin eleman sayısı kadar çalışan bir for döngüsü kurulur. Burada bu dizinin bütün elemanlarına 0 değeri atanır. İkinci olarak, synthStore dizisinin activeButtonSelectedData dizi özelliğinin elemanları için, activeButtonSelectedData dizisinin eleman sayısı kadar çalışan bir for döngüsü kurulur. Burada bu dizinin bütün elemanlarına 'false' değeri atanır. Üçüncü olarak, synthStore dizisinin activeButtonHandleData dizi özelliğinin elemanları için, activeButtonHandleData dizisinin eleman sayısı kadar çalışan bir for döngüsü kurulur. Burada bu dizinin bütün elemanlarına 0 değeri atanır. Dördüncü olarak, synthStore dizisinin monitoringValueData dizi özelliğinin elemanları için, monitoringValueData dizisinin eleman sayısı kadar çalışan bir for döngüsü kurulur. Burada bu dizinin bütün elemanlarına 0 değeri atanır. Beşinci olarak, synthStore dizisinin monitoringManualData dizi özelliğinin elemanları için, monitoringManualData dizisinin eleman sayısı kadar çalışan bir for döngüsü kurulur. Burada bu dizinin bütün elemanlarına 'false' değeri atanır. Altıncı olarak, synthStore dizisinin spectrumMinValueData dizi özelliğinin elemanları için, spectrumMinValueData dizisinin eleman sayısı kadar çalışan bir for döngüsü kurulur. Burada bu dizinin bütün elemanlarına 0 değeri atanır. Yedinci olarak, synthStore dizisinin spectrumMaxValueData dizi özelliğinin sıfır indeksli elemanına 10000 değeri atanır. Sekizinci olarak, synthStore dizisinin spectrumMaxValueData dizi özelliğinin bir indeksli elemanına 100 değeri atanır. Dokuzuncu olarak, synthStore dizisinin spectrumMaxValueData dizi özelliğinin iki indeksli elemanına 100 değeri atanır. Onuncu olarak, synthStore dizisinin signalKindData özelliğine 0 değeri atanır. On birinci olarak, synthStore

dizisinin portamentoValueData özelliğine 0 değeri atanır. On ikinci olarak, synthStore dizisinin portaButtonData özelliğine ‘false’ değeri atanır. Böylece, kullanıcının herhangi bir gridNumberSelections dizisi elemanı ile etkileşime geçerek görüntünün sahip olduğu alt çerçeve sayısını değiştirmesinin ardından, synthStore dizisinin elemanlarına yeni ilk değerlerinin atanması işlemi sonlanmış olur.

- previewbutton nesnesi: Bu nesne ile arayüzde mouse butonuna basılması ve kaldırılması ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, bu eylemin bu nesnenin üzerinde mi yapıldığı sınanır. Bu sınamanın yapılması için, mouse butonunun basılı olma durumundan çıktığı anda, bu nesnenin üzerinde olup olmadığı, bu nesnenin sahip olduğu overRect( ) fonksiyonu ile sınanır. Bu sınama ‘true’ değerini döner ise, nesnenin mouseReleased( ) yöntemi çalıştırılır. Bu yöntem içerisinde, bu nesnenin selected özelliği sınanır. Bu özellik ‘true’ ise ‘false’ değeri atanır. Eğer ‘false’ ise ‘true’ değeri atanır.
- gridbutton nesnesi: Bu nesne ile arayüzde mouse butonuna basılması ve kaldırılması ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, bu eylemin bu nesnenin üzerinde mi yapıldığı sınanır. Bu sınamanın yapılması için, mouse butonunun basılı olma durumundan çıktığı anda, bu nesnenin üzerinde olup olmadığı, bu nesnenin sahip olduğu overRect( ) fonksiyonu ile sınanır. Bu sınama ‘true’ değerini döner ise, nesnenin mouseReleased( ) yöntemi çalıştırılır. Bu yöntem içerisinde, bu nesnenin selected özelliği sınanır. Bu özellik ‘true’ ise ‘false’ değeri atanır. Eğer ‘false’ ise ‘true’ değeri atanır.
- linebutton nesnesi: Bu nesne ile arayüzde mouse butonuna basılması ve kaldırılması ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, bu eylemin bu nesnenin üzerinde mi yapıldığı sınanır. Bu sınamanın yapılması için, mouse butonunun basılı olma durumundan çıktığı anda, bu nesnenin üzerinde olup olmadığı, bu nesnenin sahip olduğu overRect( ) fonksiyonu ile sınanır. Bu sınama ‘true’ değerini döner ise, nesnenin mouseReleased( ) yöntemi çalıştırılır. Bu yöntem içerisinde, bu nesnenin

selected özelliği sınanır. Bu özellik ‘true’ ise ‘false’ değeri atanır. Eğer ‘false’ ise ‘true’ değeri atanır.

- portabutton nesnesi: Bu nesne ile arayüzde mouse butonuna basılması ve kaldırılması ile etkileşime geçildiği için, mouse butonundan parmağın çekilmesi eyleminde, bu eylemin bu nesnenin üzerinde mi yapıldığı sınanır. Bu sınamanın yapılması için, mouse butonunun basılı olma durumundan çıktığı anda, bu nesnenin üzerinde olup olmadığı, bu nesnenin sahip olduğu overRect( ) fonksiyonu ile sınanır. Bu sınama ‘true’ değerini döner ise, nesnenin mouseReleased( ) yöntemi çalıştırılır. Bu yöntem içerisinde, bu nesnenin selected özelliği sınanır. Bu özellik ‘true’ ise ‘false’ değeri atanır. Eğer ‘false’ ise ‘true’ değeri atanır. Ardından, bu portabutton nesnesinin selected özelliğinin değeri, synthStore dizisinin güncel elemanının portaButtonData özelliğine atanır.
- PortamentoTool nesnesi: Bu nesne ile arayüzde sürükleme eylemi ile etkileşim kurulduğundan ve parmağın mouse butonundan kaldırılması durumunda bu eylem gerçekleştirilemeyeceği için, PortamentoTool nesnesini tutan portamento değişkeninin selected özelliğine ‘false’ değeri atanır.
- Görüntü alt çerçevesinin seçildiği ızgaralı alan: Kullanıcının, kontrollerini geçekleştirmek üzere görüntü alt çerçeveleri için seçim yapabileceği ızgaralı alanda herhangi bir seçim yapıp yapmadığının sınanması da mouseReleased( ) fonksiyonu içerisinde gerçekleşir. Eğer kullanıcı, bu seçim alanında bulunan ve toplam çerçeve sayısına göre belirlenmiş ızgaralarla bölünen alanda bir hücre üzerine gelip mouse butonu ile seçim yapar ise, bu yeni seçimin, yani güncel alt çerçevenin, bilgisi global cRow değişkenine atanır.

  Bu işlevlerin gerçekleşeceği bu alan, GraphicUserInterface( ) fonksiyonu içerisinde, processing ortamının standart rect( ) ve line( ) fonksiyonları kullanılarak oluşturulmuştur. Bu nedenle, bu alan elemanları için, nesnelerini yaratacağımız bir sınıf tanımlanmamıştır. Sınamalar, mouseReleased( ) fonksiyonu içerisinde gerçekleştirilir.

Öncelikle kullanıcının mouseReleased( ) fonksiyonunu çalıştıran eyleminin, mouse işlecinin ızgaralı seçim alanının üzerinde yapılıp yapılmadığının sınanması yapılır. Eğer bu sınama 'true' olursa, ilgili seçim alanının kendi iç sınamalarına geçilir.

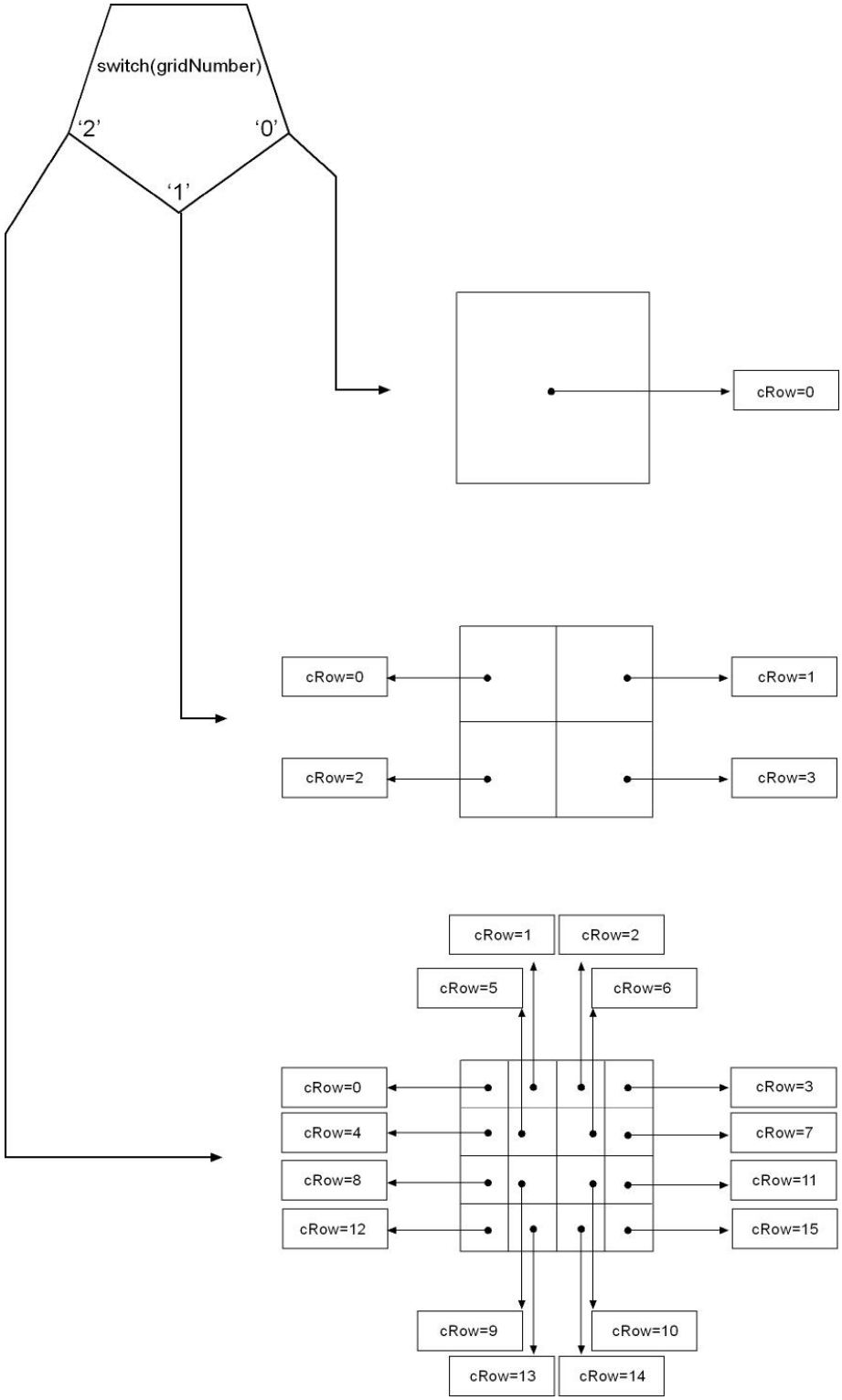
İlk olarak, gridNumber global değişkeni sınanır. Çünkü bu değişken görüntü alanının kaç alt çerçeveye bölündüğünün bilgisini taşır. Buna göre ızgaralı alanın hücrelerinin sayısı değişecektir.

Global gridNumber değişkeni 0 ise, tek bir görüntü çerçevesi olduğu için, cRow değişkenine 0 değeri atanır.

Global gridNumber değişkeni 1 ise, toplam dört adet alt çerçeve vardır ve bunun doğrultusunda, cRow değişkeni 0'dan 3'e kadar değer alabilir. Bu durumda öncelikle, mouse işlecinin yatay konumunun, ızgaralı alanın hangi sütununda olduğu sınanır. Sütun bilgisini taşıyan yerel değişken 0 ya da 1 değerini alır. Ardından mouse işlecinin dikey konumunun, ızgaralı alanın hangi satırında olduğu sınanır. Satır bilgisini taşıyan yerel değişken 0 ya da 1 değerini alır. Son olarak, satır bilgisini taşıyan yerel değişkenin iki katı ile sütun bilgisini taşıyan yerel değişken toplanır. Çıkan sonuç global cRow değişkenine atanır.

Global gridNumber değişkeni 2 ise, toplam on altı adet alt çerçeve vardır ve bunun doğrultusunda, cRow değişkeni 0'dan 15'e kadar değer alabilir. Bu durumda öncelikle, mouse işlecinin yatay konumunun, ızgaralı alanın hangi sütununda olduğu sınanır. Sütun bilgisini taşıyan yerel değişken 0, 1, 2, 3 değerlerinden birini alır. Ardından mouse işlecinin dikey konumunun, ızgaralı alanın hangi satırında olduğu sınanır. Satır bilgisini taşıyan yerel değişken 0, 1, 2, 3 değerlerinden birini alır. Son olarak, satır bilgisini taşıyan yerel değişkenin dört katı ile sütun bilgisini taşıyan yerel değişken toplanır. Çıkan sonuç global cRow değişkenine atanır.

Böylece kullanıcının ızgaralı alanda hangi hücreyi seçtiği belirlenerek, bunun bilgisi global cRow değişkenine atanmış olmaktadır. (Şekil 3.48)



Şekil 3.48 Kullanıcı kontrollerine göre, ızgaralı alanda hücrelerin seçimine göre cRow global değişkeninin alacağı değerler

Bu işlemler bitirildikten sonra, mouseReleased( ) fonksiyonu içerisinde yapılması gereken bir diğer işlem, LoadValues( ) fonksiyonun çalıştırılmasıdır. Bu fonksiyonun görevi, synthStore dizisinin güncel olan, yani o anda kullanıcı arayüzünde görünen, ses kanalı ile ilgili olarak bir değişiklik yapıldığında bu değişikliğin arayüz elemanlarına aktarılması işlemidir. Burada gerçekleşen olay, synthStore dizisinin ilgili indekli elemanının, yani yeni cRow ve cCol değişkeninin gösterdiği elemanının, özelliklerinin sahip olduğu değerlerin, arayüz elemanlarına aktarılmasıdır. Bunun için, mouseReleased( ) fonksiyonun sonunda LoadValues( ) fonksiyonu çalıştırılır. Bu fonksiyonda sırasıyla şu işlemler gerçekleştirilir.

İlk olarak; synthStore dizisinin cRow ve cCol indeksi elemanının activeButtonValueData dizi özelliğinin değerleri, activeButtons dizisinin karşılıklı elemanlarının value değişkenlerine aktarılır.

İkinci olarak; synthStore dizisinin cRow ve cCol indeksi elemanının activeButtonSelectedData dizi özelliğinin değerleri, activeButtons dizisinin karşılıklı elemanlarının selected değişkenlerine aktarılır.

Üçüncü olarak; synthStore dizisinin cRow ve cCol indeksi elemanının activeButtonHandleData dizi özelliğinin değerleri, activeButtonHandles dizisinin karşılıklı elemanlarının value değişkenlerine aktarılır.

Dördüncü olarak; synthStore dizisinin cRow ve cCol indeksi elemanının monitoringValueData dizi özelliğinin değerleri, monitorings dizisinin karşılıklı elemanlarının soundValue değişkenlerine aktarılır.

Beşinci olarak; synthStore dizisinin cRow ve cCol indeksi elemanının spectrumMinValueData dizi özelliğinin değerleri, spectrums dizisinin karşılıklı elemanlarının minSpectrum değişkenlerine aktarılır.

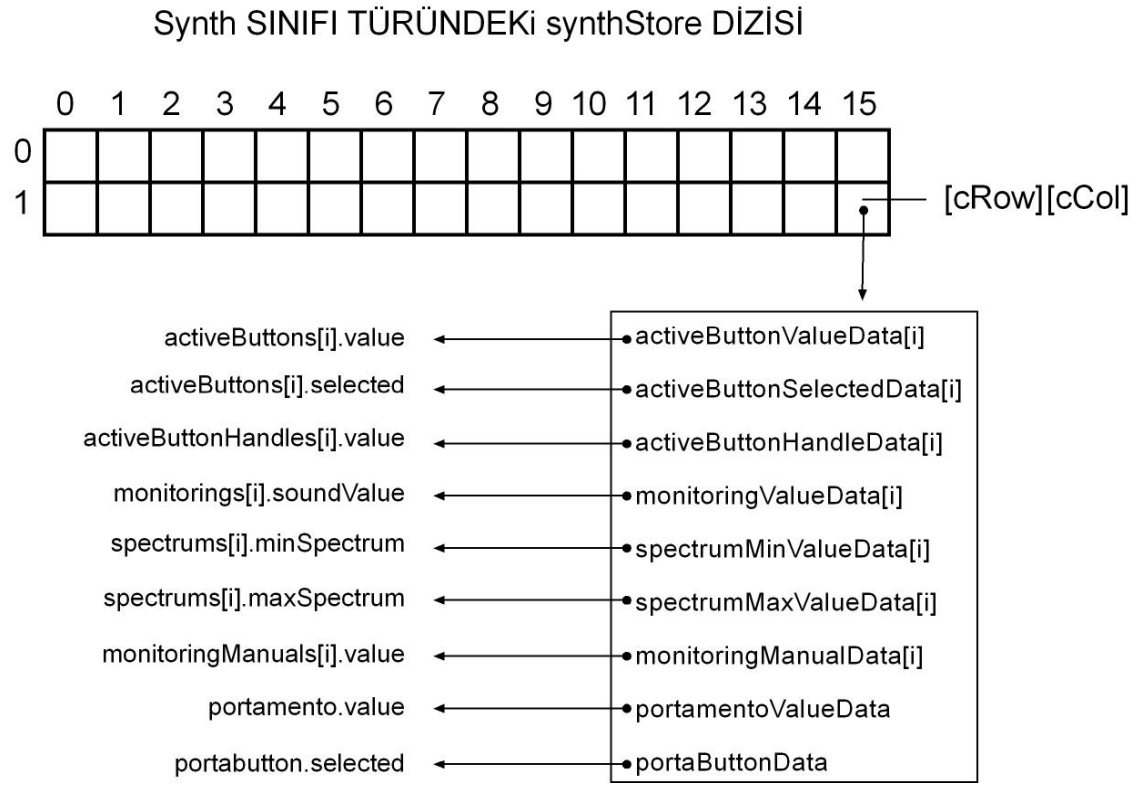
Altıncı olarak; synthStore dizisinin cRow ve cCol indeksi elemanının spectrumMaxValueData dizi özelliğinin değerleri, spectrums dizisinin karşılıklı elemanlarının maxSpectrum değişkenlerine aktarılır.

Yedinci olarak; synthStore dizisinin cRow ve cCol indeksi elemanının monitoringManualData dizi özelliğinin değerleri, monitoringManuals dizisinin karşılıklı elemanlarının value değişkenlerine aktarılır.

Sekizinci olarak; synthStore dizisinin cRow ve cCol indeksi elemanının portamentoValueData özelliğinin değeri, portamento değişkeninin value özelliğine aktarılır.

Dokuzuncu olarak; synthStore dizisinin cRow ve cCol indeksi elemanının portaButtonData özelliğinin değeri, portabutton değişkeninin selected özelliğine aktarılır.

LoadValues( ) fonksiyonu sonlandırıldıktan sonra, kullanıcının seçtiği yeni görüntü çerçevesine ait olan bütün ses kanalı özelliklerinin değerlerinin arayüzde bulunan nesnelere aktarılması işlemi bitirilmiş olur. (Şekil 3.49)



Şekil 3.49 Kullanıcının alt çerçeve seçimine göre synthStore dizisinin güncel indeksli değerlerinin arayüz nesnelerine aktarılması

Böylece, mouseReleased( ) fonksiyonunun sonlanmasıyla, sistem işleyişi tamamlanmış olur. Bundan sonraki bölümde, kullanıcı kontrollerinin arka planda işleyen sistemi yönlendirilmesini sağlayan kullanıcı grafik arayüzü incelenecektir.

3.4 ARACIN ARAYÜZÜ

3.4.1 Kullanıcı Arayüzünün Genel Özellikleri

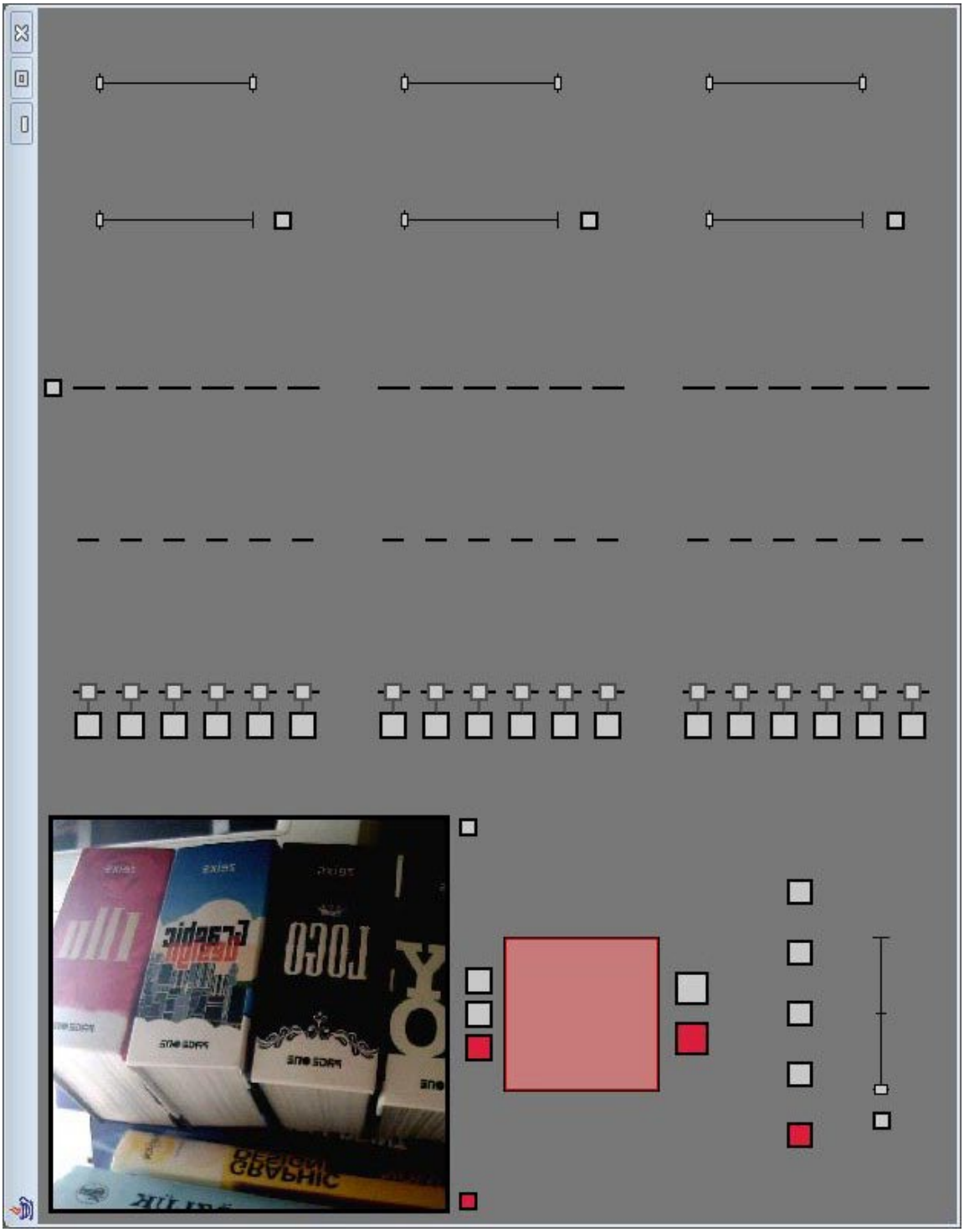
Kullanıcının işitsel-sanat aracı ile olan etkileşimlerinin gerçekleşmesi için gerekli olan arayüz, sistemin işlevlerinin oluşturduğu değerlerin gösterilmesini ve bu işlevlerin yönlendirilmesini sağlayacak özelliklere sahip olacak şekilde tasarlanmıştır.

Burada belirtilmesi gereken önemli bir nokta, çalışmanın GraphicUserInterface( ) fonksiyonunun anlatıldığı kısımda gösterildiği gibi, kullanıcı arayüz elemanlarının oluşturulması sırasında, dışarıdan hiçbir imaj alınmaması ve sadece processing ortamının sağladığı standart fonksiyon ve özelliklerinin kullanılması ile minimal arayüz elemanları oluşturulmasıdır. Buradaki amaç, aracın temel etkileşimlerinin ve işleyişinin gösterilmesidir. Tasarladığımız işitsel-sanat aracı, düzenek olarak tasarlanmış ve arayüz işlevleri gerçekleştirilmiştir. Bu noktadan sonra, arayüz için kavramsal bir temelde oluşturulacak olan grafik çerçeve farklı bir çalışmanın konusu olacaktır.

Kullanıcı grafik arayüzünün arka plan rengi, nesnelerin boyutlarının ve yerlerinin belirlenmesi ve etkileşim için gerekli olan güncelleme yöntemleri ile ilgili olarak, çalışmanın 3.3.3 no'lu 'Kullanıcı Grafik Arayüzünün Yaratılması' başlıklı kısımda teknik bilgiler verildi. Burada aracın oluşturulma amacına ulaşılmasını ve tasarlanan etkileşimleri imkanlı duruma getiren arayüz ve elemanları incelenecektir.

3.4.2 Aracın İşlevleri ve Arayüz Elemanları

Arayüz, bütün işlevleri gerçekleştirecek olan nesnelere, yatayda 800 piksel ve dikeyde 600 piksel boyutlarında olan bir alan içerisinde sahiptir. (Şekil 3.50)



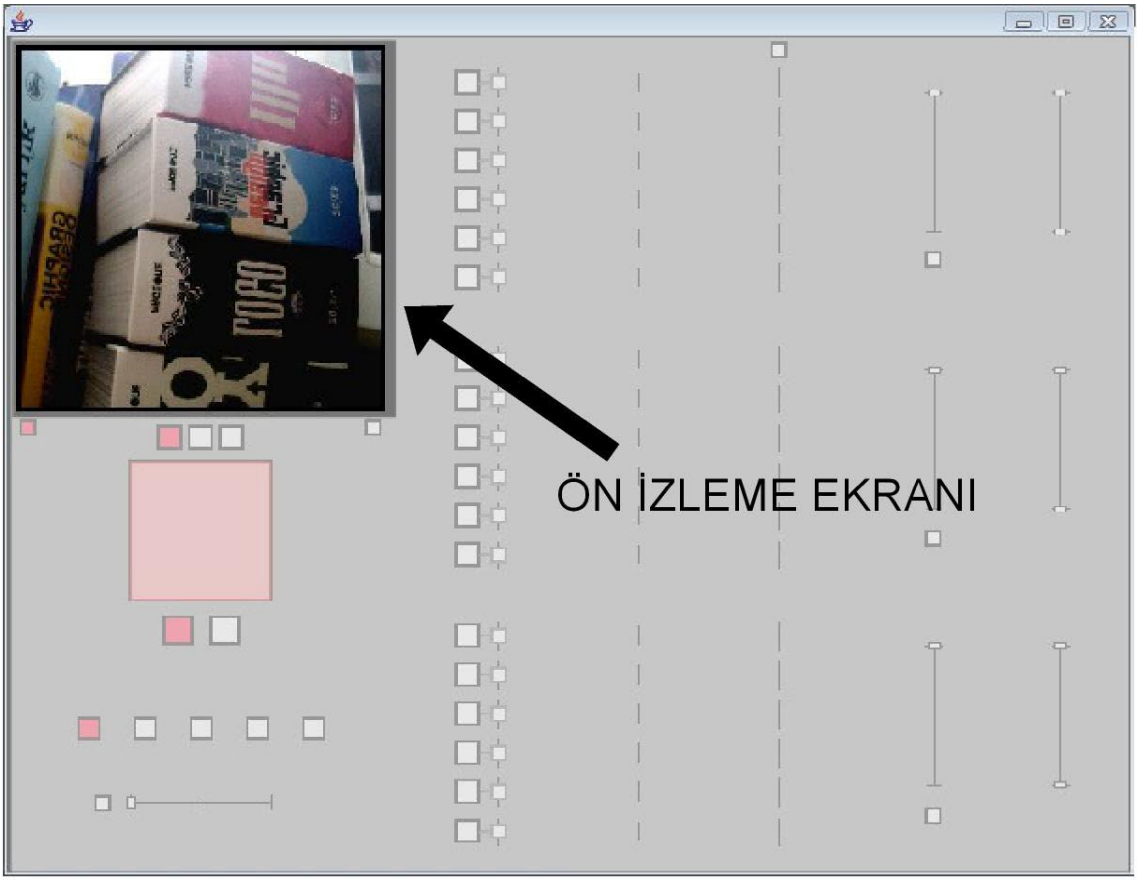
Şekil 3.50 İşitsel-sanat aracının kullanıcı arayüzünün başlangıç değerleri ile görünüşü

Arayüz içerisindeki bütün elemanlar, gri-120 değerindeki arkaplan rengi üzerinde bulunurlar. Bu nesneler özellikleri ile ilgili olarak işlevsel bir şekilde yerleştirilmişlerdir. Kullanıcının aracı kullanırken karşılaşacağı ihtiyaçlar ve ilişkiler göz önünde tutularak bir sıra düzeni oluşturulmuştur. Arayüz içerisinde bulunan

nesneler; görüntülerin izleneceği ön izleme ekranı, ön izleme ekranının açılıp kapanmasını sağlayan buton, ön izleme ekranında ızgaraların görünmesini sağlayan buton, görüntü kaynağının kaç alt çerçeveye bölüneceğini belirleyen butonlar, görüntünün alt çerçevesinin seçilmesini sağlayan seçim alanı, alt çerçeveye bağlı iki ses kanalı arasında seçim yapılmasını sağlayan butonlar, ses sinyal dalgası türünü belirleyen butonlar, ses kanallarının portamento değerini açıp kapayan buton, ses kanallarının portamento değerlerini belirleyen kulp, ses kanallarına aktarılacak olan görüntü özelliklerinin açılıp kapanmasını sağlayan butonlar, görüntü özelliklerinin oranlarını belirleyen kulplar, ses sinyal özelliklerinin değerlerini gösteren ve belirleyen kulplar, ses sinyal özelliklerinin spektrumlarını gösteren ve belirleyen kulplar, ses sinyal değerlerinin kaynağını belirleyen kulplar ve görüntü ile ses özellikleri arasında çizilecek olan çizgilerin açılıp kapanmasını sağlayan kulptur.

3.4.2.1 Kameradan alınan görüntünün ön izleme ekranında gösterilmesi

Kameradan alınan görüntülerin kullanıcı tarafından izlenmesi ve bunun sonucunda oluşan ses değerlerine göre yeniden değerlendirilerek tekrar görüntü çerçevelerinin belirlenmesi için, kameradan alınan görüntülerin ön izlenmesinin yapıldığı bir alana ihtiyaç vardır. Bu alan aracın arayüzünde sol üst köşede yer alır. İçindeki görüntünün diğer alanlardan ayrılması için siyah bir çerçeveye sahiptir. Bu alan 256 piksel yatay ve 256 piksel dikey boyutundadır. Aracın başlangıç değerlerinin, ön izleme alanın gözükmesi ve ızgaraların gözükmemesi yönünde olmasından dolayı, başlangıçta kameradan gelen görüntü direk yansıtılarak başlar ve herhangi bir ızgara ile çerçevelenmemiş olarak görülür. (Şekil 3.51)



Şekil 3.51 Ön izleme ekranı

3.4.2.2 Ön izleme ekranının açılıp kapanması

Kullanıcı ön izleme ekranına kameradan gelen görüntülerin yansıtılıp yansıtılmamasına karar verebilir. Bu kararın sisteme tanıtılması için ön izleme ekranının sol altında bulunan buton kullanılır. Bu buton başlangıç değeri olarak seçili durumda olur. Bu yüzden içi kırmızı renktedir ve ön izleme ekranında da görüntü vardır. (Şekil 3.52)

Şekil 3.52 Ön izlemenin yapılıp yapılmayacağının gösterildiği buton seçili durumda

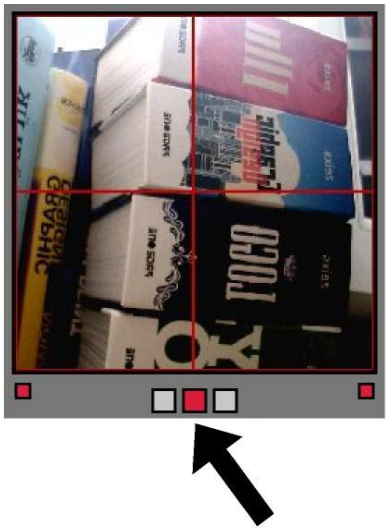
Ancak kullanıcı, bu butona basarak ön izleme ekranında görüntü olmasını engelleyebilir. Bu durumda, ön izleme ekranında herhangi bir görüntü olmayacak ve gri bir alan olarak görülecektir. İlgili buton da, seçili olmama halini göstermek için gri renk dolguya sahip olacaktır. (Şekil 3.53)

Şekil 3.53 Ön izlemenin yapılıp yapılmayacağının gösterildiği buton seçili olmayan durumda

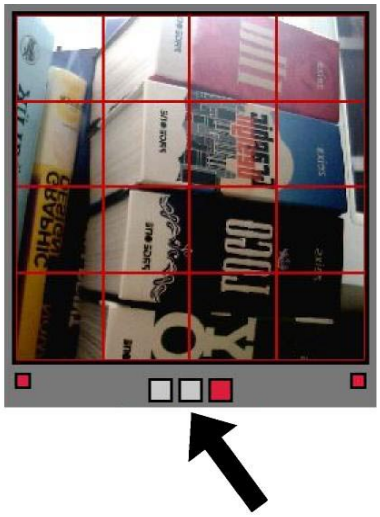
3.4.2.3 Görüntülerin alt çerçevelere bölünmesi

Kullanıcı, kameradan gelen görüntüleri kaç alt çerçeveye böleceğinin bilgisini üç adet buton içerisinde seçim yaparak sisteme bildirir. Bu butonlar, ön izleme çerçevesinin hemen altında yer alırlar. Bu üçlü gruptan, soldaki buton tek çerçeveyi, ortadaki buton dört alt çerçeveyi, sağdaki buton on altı alt çerçeveyi temsil eder. Aracın başlangıç değeri olarak, tek çerçeveli ekran seçili olduğu için, bu butonlardan solda olanı seçili durumda görülür. Kullanıcı, kameradan gelen görüntüyü dört alt çerçeveye bölmek isterse, bu durumda ortadaki butonu seçmesi gerekmektedir. Eğer, on altı alt çerçeveye bölmek isterse, sağdaki butonu seçmesi gerekmektedir. Hangi buton seçili durumda ise, bu butonun seçili olduğunu göstermek için dolgu rengi kırmızı, diğer butonların dolgu rengi seçili olmadıklarını göstermek için gri olacaktır. (Şekil 3.54, Şekil 3.55, Şekil 3.56)

Şekil 3.54 Görüntünün tek çerçeveli olarak kullanılması



Şekil 3.55 Görüntünün dört alt çerçeveli olarak kullanılması



Şekil 3.56 Görüntünün on altı alt çerçeveli olarak kullanılması

3.4.2.4 Ön izleme ekranında ızgaraların gösterilmesi

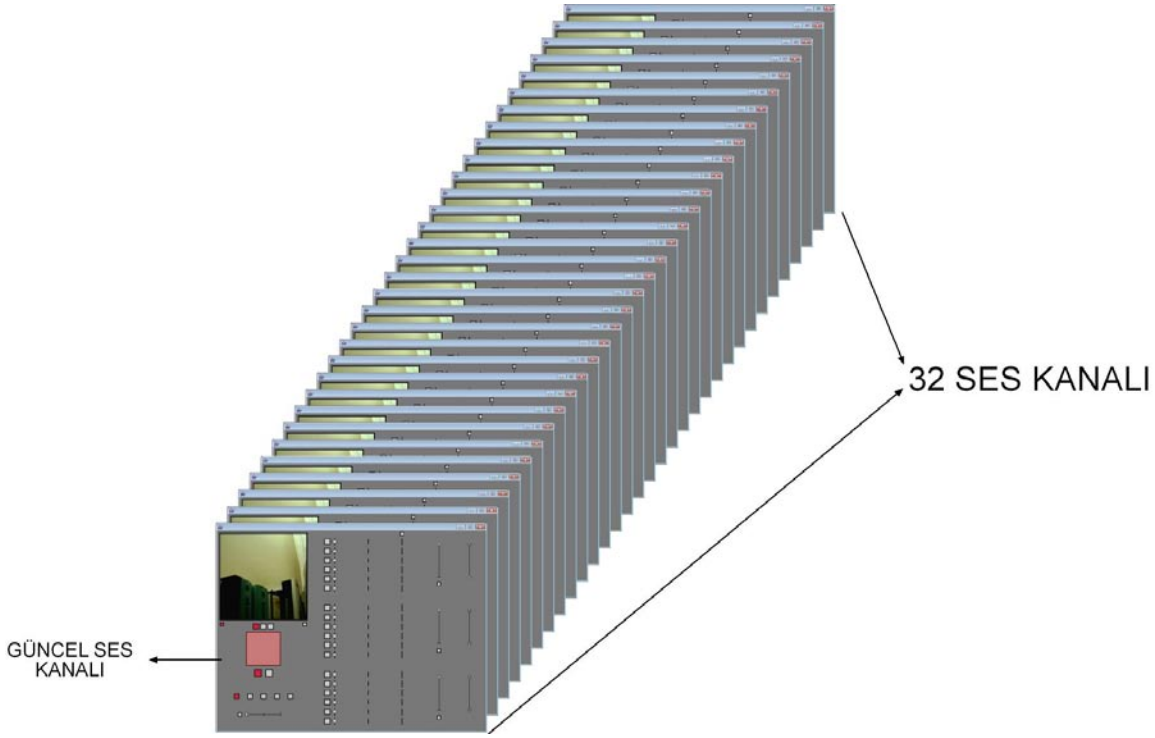
Kullanıcı ön izleme ekranının alt çerçevelerini daha rahat görebilmek için, bu alanı ızgaralı bir şekilde görmek isteyebilir. Bu durumda, ön izleme ekranının sağ alt köşesinde bulunan butonu seçmesi gerekmektedir. Bu buton, araç tek çerçeveli olarak çalışmaya başladığı için, başlangıç değeri olarak seçili olmama değerine sahiptir.

Kullanıcı istediği takdirde, bu ızgaraları açarak görüntüleri daha rahat kontrol edebilir. (Şekil 3.57)

Şekil 3.57 Ön izleme ekranında ızgaraların görülmesini kontrol eden butonun seçili olma ve seçili olmama durumu

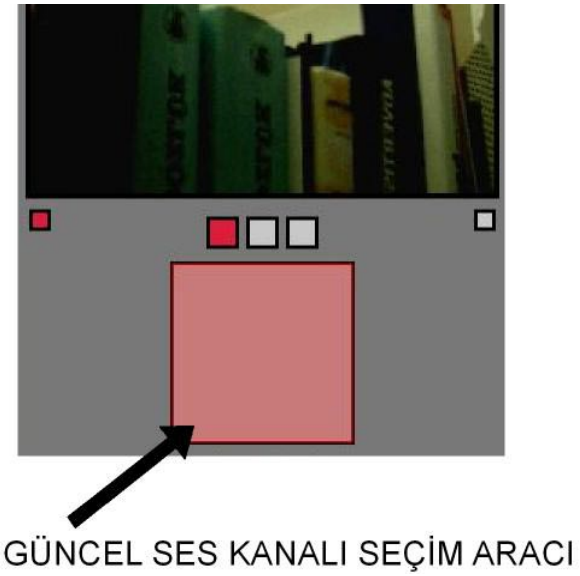
3.4.2.5 Güncel görüntü alt çerçevesinin seçilmesi

Tasarladığımız araç, aynı anda otuz iki ses kanalını çalıştırma kapasitesinde olduğu için, kullanıcı bu kanallardan güncel olarak hangisi ile ilgilendiğini belirtmek durumundadır. Böylece arayüzde gösterilen değerlerin hangi ses kanalına ait olduğunun belirlenmesi gerekmektedir. Eğer kullanıcı, tek görüntü çerçevesi seçti ise iki ses kanalı, dört görüntü çerçevesi seçti ise sekiz ses kanalı, on altı görüntü çerçevesi seçti ise otuz iki ses kanalı üst üste aynı anda çalışacaktır. (Şekil 3.58)



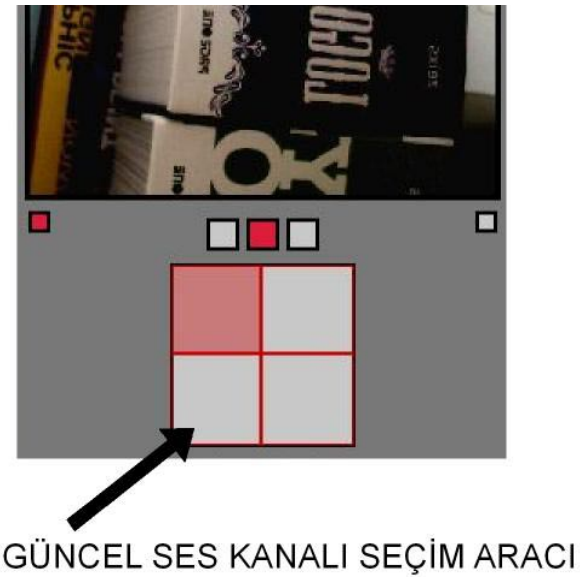
Şekil 3.58 Kullanıcının on altı görüntü kanalını seçmesi durumunda aynı anda çalışan ses kanalları ve güncel ses kanalı

Ancak bu ses kanallarından sadece bir tanesinin değerleri arayüzde gösterilecektir. Bu güncel kanalın belirlenmesi, arayüzde bulunan seçim aracı ile yapılır. Bu araç, kullanıcının görüntüyü kaç alt çerçeveye böldüğüne göre değişik değerler alır. Eğer kullanıcı sadece tek bir çerçevede çalışmak istiyorsa, burada seçim yapılacak bir hücre olmadığı için, bu seçim alanında sadece bir tane hücre görülür ve bu hücre aktif olduğundan dolayı, içi kırmızıya boyanır. (Şekil 3.59)



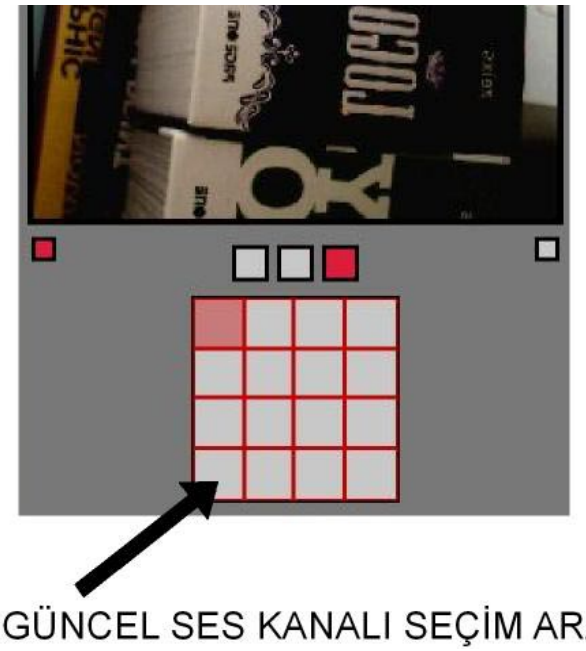
Şekil 3.59 Kullanıcının tek çerçeveli görüntü seçmesi durumunda seçim alanının görünümü

Eğer kullanıcı, dört çerçeveli bir görüntü seçerse, seçim aracı toplam dört hücre barındıracak şekilde ızgaralarla bölünür. Kullanıcının seçtiği görüntü çerçevesinin içi kırmızıya boyanırken, seçili olmayan diğer hücrelerin içi gri renk ile boyanır. (Şekil 3.60)



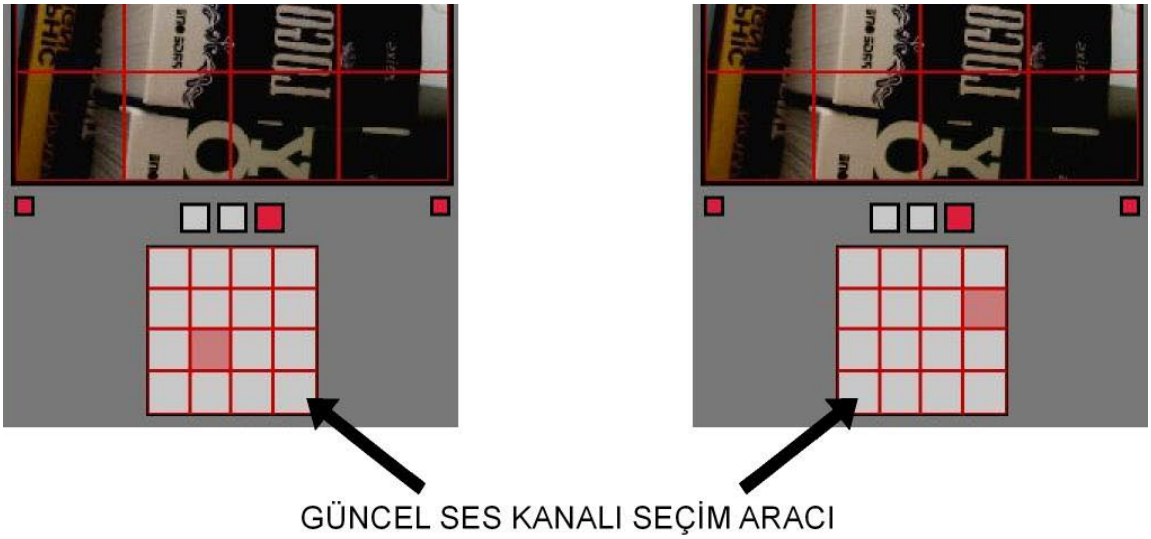
Şekil 3.60 Kullanıcının dört çerçeveli görüntü seçmesi durumunda seçim alanının görünümü

Eğer kullanıcı, on altı çerçeveli bir görüntü seçerse, seçim aracı toplam on altı hücre barındıracak şekilde ızgaralarla bölünür. Kullanıcının seçtiği görüntü çerçevesinin içi kırmızıya boyanırken, seçili olmayan diğer hücrelerin içi gri renk ile boyanır. (Şekil 3.61)



Şekil 3.61 Kullanıcının on altı çerçeveli görüntü seçmesi ile seçim alanının görünümü

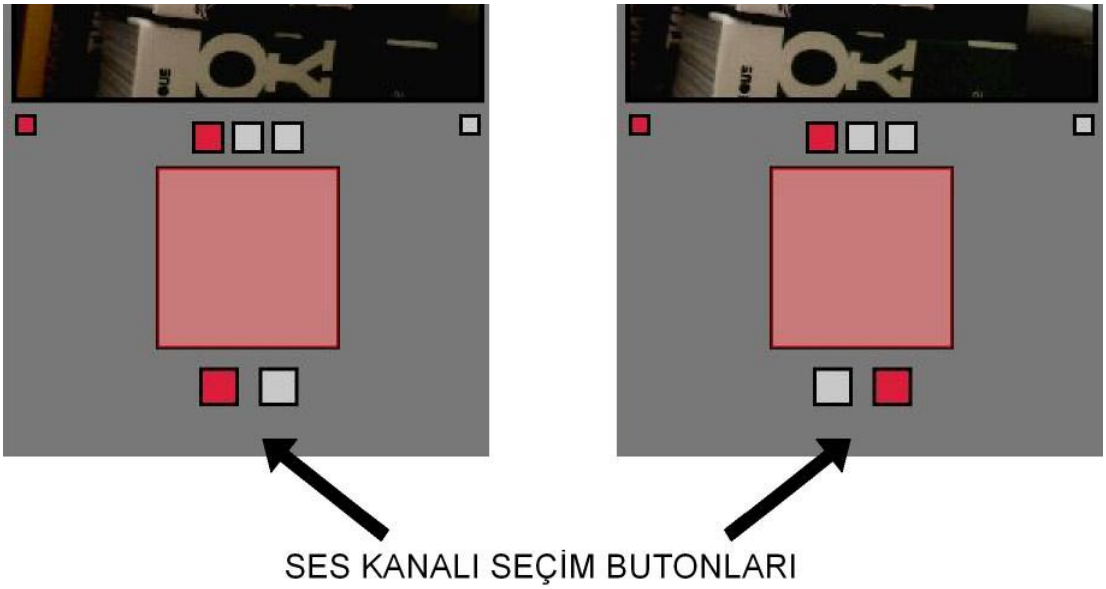
Burada görülen seçimler, aracın başlangıç değeri olan ilk görüntü kanalının seçimini göstermektedir. Kullanıcı diğer hücreleri seçerek, başka görüntü kanallarını aktif hale getirebilir. (Şekil 3.62)



Şekil 3.62 Kullanıcının on altı çerçeveli görüntü alanında farklı hücreleri seçmesi durumunda bu seçim alanının aldığı görünümler

3.4.2.6 Görüntü çerçevesine bağlı ses kanalının belirlenmesi

Kullanıcının sisteme hangi ses çıkış kanalı ile ilgilendiğini kesin olarak belirtebilmesi için, ilgilendiği görüntü hücresini belirledikten sonra, bu görüntü hücresinden kaynaklanan iki ses kanalından birini seçmesi gerekir. Her bir görüntü hücresinden kaynaklanan iki ses kanalı için yapılacak olan seçim, görüntü hücresi seçim alanının altında bulunan iki adet buton ile yapılır. Bu butonların her biri, bir ses kanalı çıkışını temsil eder. Hangi buton seçili ise, içi kırmızı dolgu rengi ile boyanır. Diğer buton ise, seçili olmama halini göstermek için gri dolgu rengi ile boyanır. (Şekil 3.63)

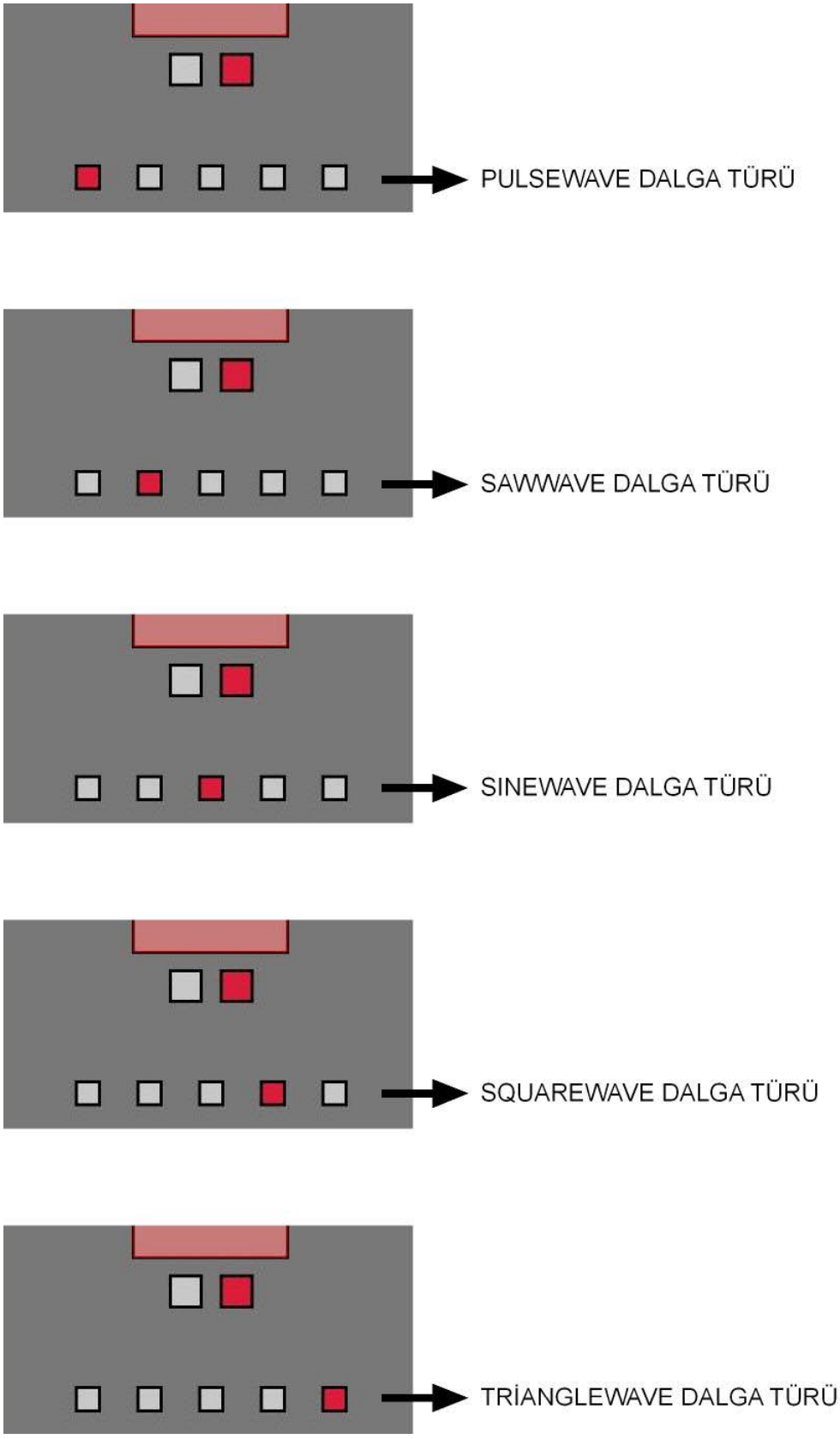


Şekil 3.63 Tek bir görüntü hücresine bağlı ses kanallarından birinin belirlenmesi durumunda butonların görünümü

Bu iki ses kanalı da, aynı görüntü çerçevesinden beslenir. Değerlerini belirleyen görüntü parametreleri aynı olur. Ancak kullanıcının bu görüntü değerlerini farklı birleştirmelerle, farklı ses sinyal sonuçları elde etmesi sağlanır. Kullanıcı her bir görüntü hücresinden, en fazla iki ses kanalı yaratabilir.

3.4.2.7 Ses sinyal dalgası türünün belirlenmesi

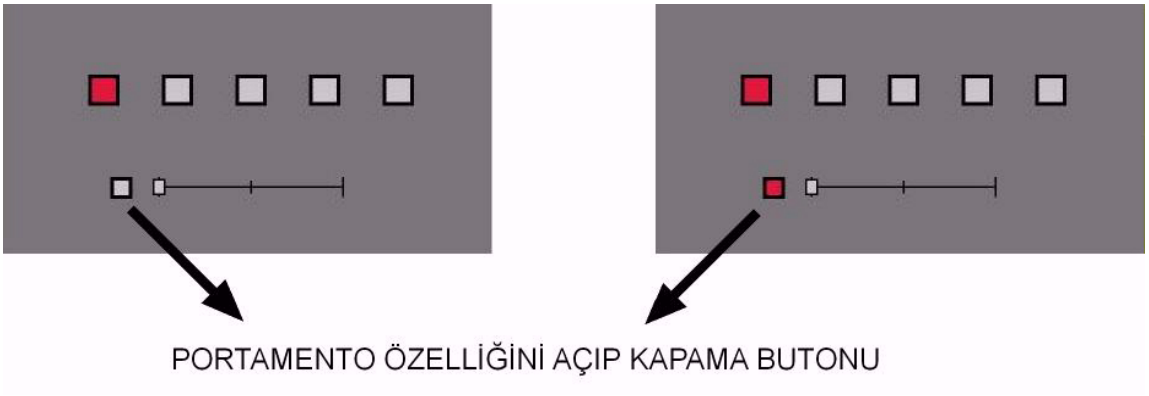
Burada kullanıcı kontrolü olarak, güncel ses kanalının ses sinyal türü seçilmektedir. Kullanıcı beş farklı ses sinyal türünden birini seçmek durumundadır. Bu beş sinyal türü, arayüzde yan yana dizilmiş beş buton ile temsil edilir. Birinci buton pulsewave dalgalarını, ikinci buton sawwave dalgalarını, üçüncü buton sinewave dalgalarını, dördüncü buton squarewave dalgalarını, beşinci buton trianglewave dalgalarını temsil eder. Kullanıcı güncel ses kanalı için bu dalgalardan sadece tek bir tanesini seçebilir. (Şekil 3.64)



Şekil 3.64 Ses dalgası türünü belirleyen butonlar

3.4.2.8 Portamento özelliğinin açılıp kapanması

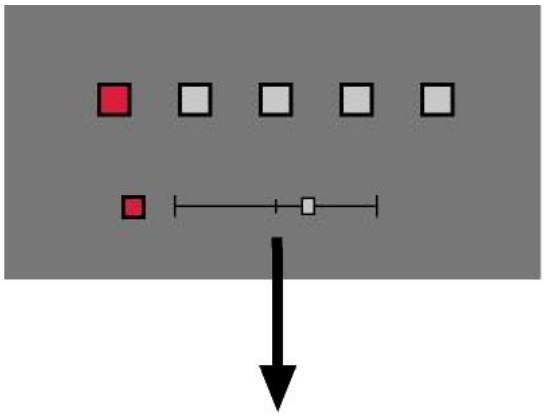
Ses sinyal kanallarının portamento değerlerini değiştirebilir. Başlangıç değeri olarak, hiçbir ses sinyal kanallının portamento değeri yoktur. Kullanıcı güncel ses kanalının portamento değerini değiştirebilmek için, öncelikle ses kanalının portamento özelliğini açması gerekir. Bunun için arayüzde bulunan ve ses sinyal türlerini belirleyen buton grubunun altında bulunan portamento butonunu seçmelidir. Bu buton seçili olduğunda, kullanıcı ses sinyal kanalının portamento değerini değiştirebilme hakkına sahip olur. (Şekil 3.65)



Şekil 3.65 Portamento butonunun açık ve kapalı olma durumundaki görünümü

3.4.2.9 Portamento değerinin belirlenmesi

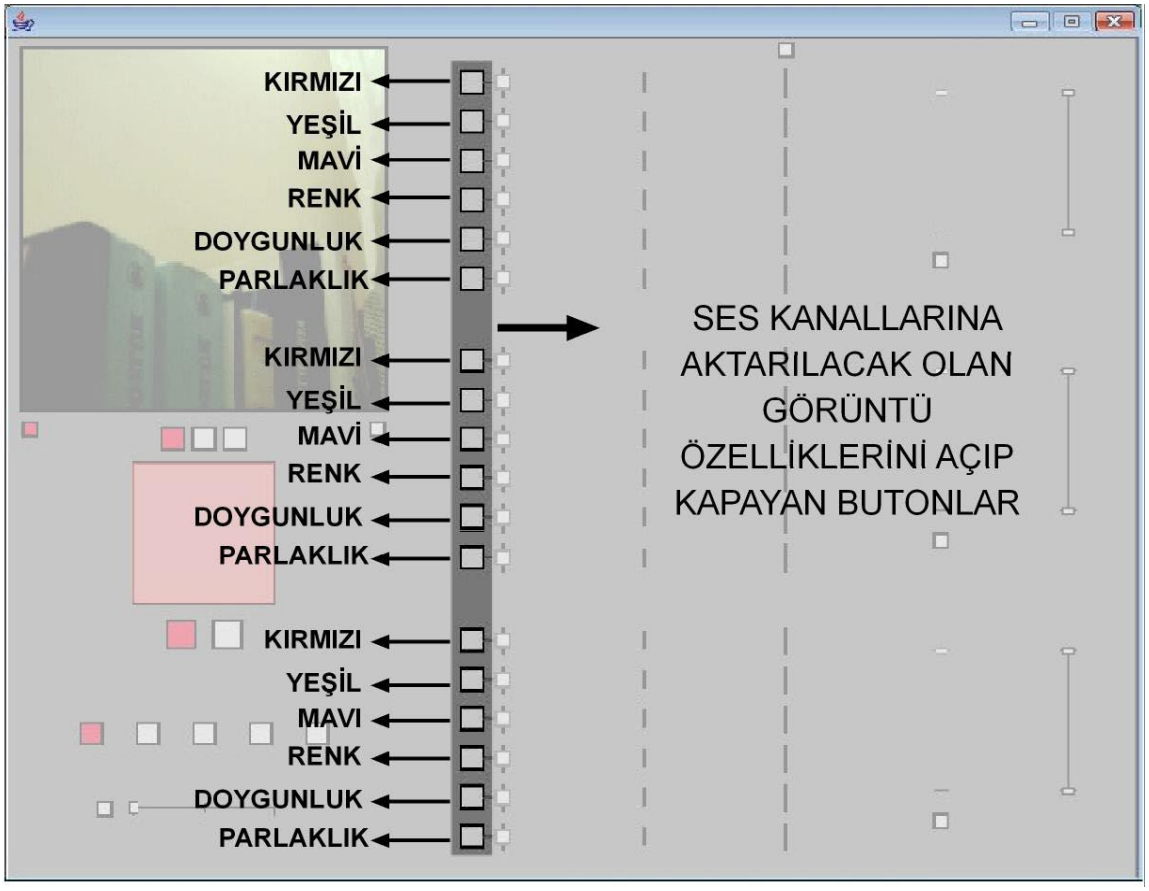
Kullanıcı, güncel ses kanalı için portamento özelliğini açtıktan sonra bu değerin niceliğini belirleme hakkına sahip olur. Bu portamento niceliği, arayüzde portamento butonunun sağında bulunan sürgülü kulp ile sağlanır. Bu kulp sağ-sol doğrultusunda hareket eder. Bu kulp en sol konumda iken, 0 milisaniye, en sağ konumda iken, 2000 milisaniye değeri taşır. Kullanıcı, aracın çalışma süreci boyunca bu değeri kulpu sürükleyerek değiştirebilir. (Şekil 3.66)



Şekil 3.66 Portamento değerını değıştıren kulpun değer taşırken aldığı görünüm

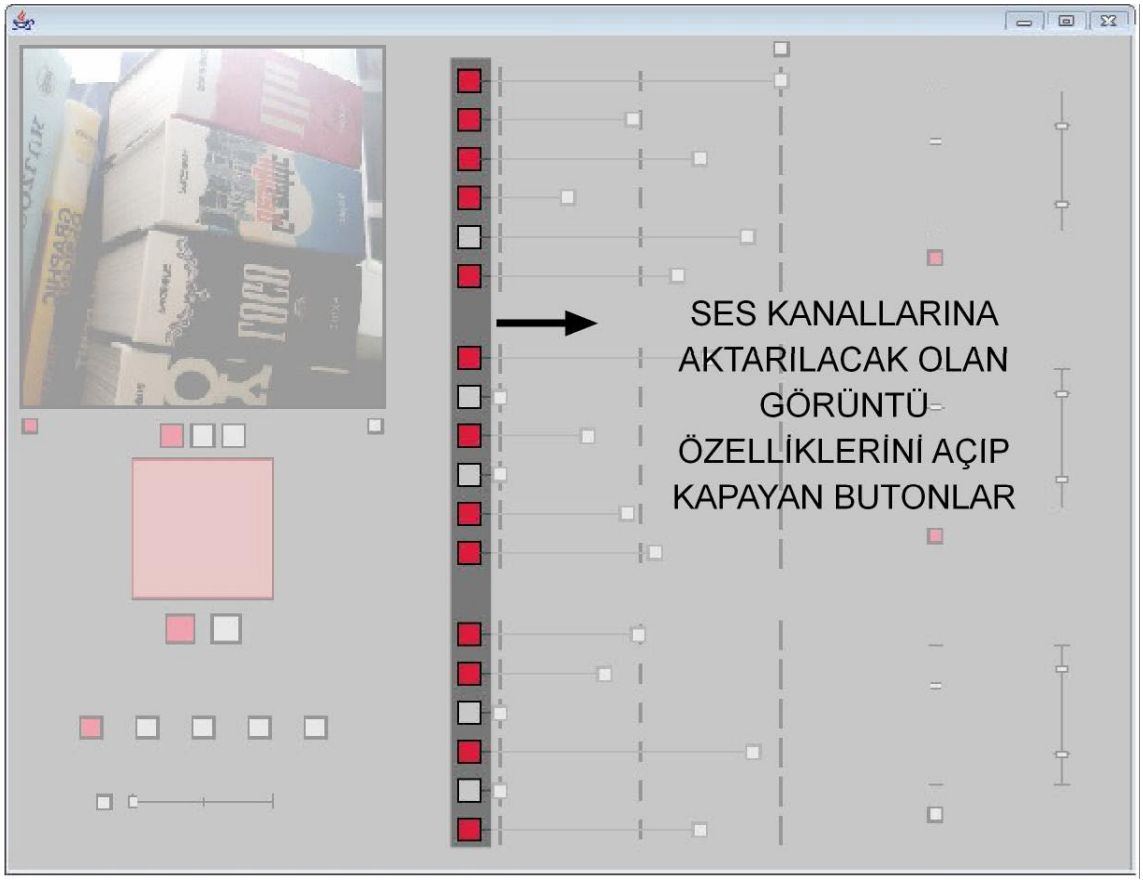
3.4.2.10 Ses kanallarına aktarılacak olan görüntü özelliklerinin açılıp kapanması

Kullanıcı, kameradan alınan görüntü değerlerinden üretilen parametrelerden hangilerinin, karşılığı olan ses sinyal özelliğinin niceliğine katılacağını belirleme kontrol hakkına sahiptir. Bu kontroller, arayüzde bulunan ve her biri görüntülerden üretilen değerler ile ilgili olan butonların açılıp kapanması ile gerçekleşir. Arayüzde toplam on sekiz adet bulunan bu butonlar ses kanallarının değerleri üretilirken bunlara hangi görüntü değerlerinin katılacağını belirlerler. Ses frekans değeri, ses yüksekliği ve pan değeri olmak üzere toplam üç adet ses özelliğinin her biri, altı adet (kırmızı, yeşil, mavi, renk, doygunluk, parlaklık) görüntü değeri ile oluşturulduğu için, arayüzde toplam on sekiz adet kontrol butonu vardır. (Şekil 3.67)



Şekil 3.67 Ses kanallarına aktarılacak olan görüntü özelliklerinin açılıp kapanmasını sağlayan butonların başlangıç görünümleri

Kullanıcı bu butonları seçerek, görüntü özelliklerine bağlı ses kanalı değerlerinin oluşturulması için gerekli olan kontrolleri sağlar. Bu butonlar yukarıdaki şekilde görüldüğü gibi, alt alta sıralanmış altışarlı gruplar halindedirler. Her altılı grup, bir ses özelliğine denk düşer. İlk altılı grup, ses frekans değeri için belirleyicidir. İkinci altılı grup, ses yüksekliği için belirleyicidir. Üçüncü altılı grup, sesin pan değeri için belirleyicidir. Kullanıcı her bir ses değeri için, bu butonlar aracılıyla kontroller sağlayarak istediği görüntü özelliklerini aktarabilir. (Şekil 3.68)



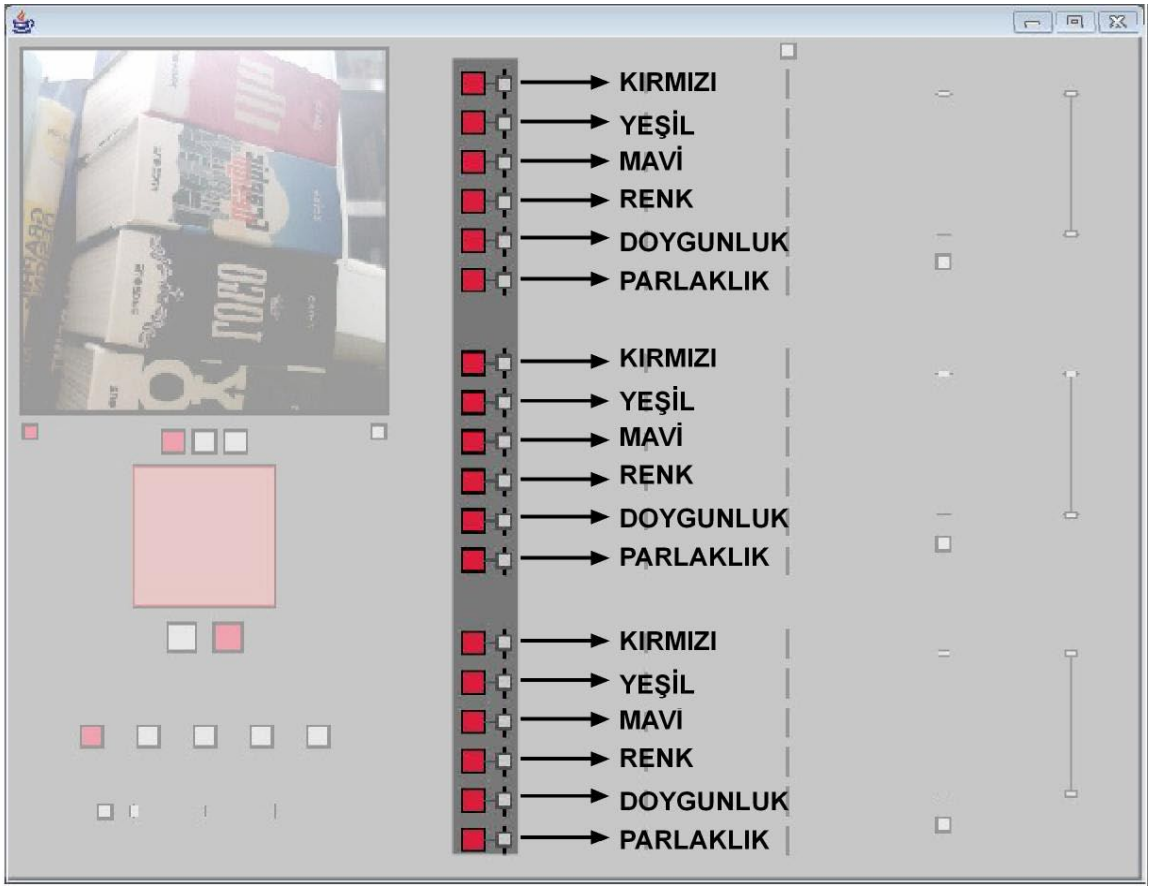
Şekil 3.68 Kullanıcının kontrolleri doğrultusunda ses değerlerine aktarılacak olan görüntü özelliklerinin belirlenmesi

Yukarıdaki şekilde görüldüğü gibi, kullanıcı birbirinden bağımsız olarak, farklı ses değerleri için farklı görüntü özelliklerini aktif hale getirebilir. Aktif hale gelen görüntü özelliklerini temsil eden butonlar, seçili olma halini gösterebilmek için dolgu rengi olarak kırmızıya boyanırlar. Diğer butonlar ise, seçili olmama halini vurgulamak için dolgu rengi olarak griye boyanırlar.

3.4.2.11 Görüntü özelliklerinin oranlarının belirlenmesi

Kullanıcı, görüntü kaynaklı olarak üreteceği ses kanalı özellikleri için hangi görüntü özelliklerini aktif hale getireceğini belirledikten sonra, bu özelliklerin değerlerinin hangi oranda birleştireceğini belirlemek durumundadır. Her bir görüntü özelliği butonunun yanında, yine bu butona bağlı olan ve oranları belirlemeye yarayan kulplar vardır. Bu kulplar sağ-sol doğrultusunda sürüklenerek kullanılır. Bütün kulpların en sol konumu yüzde 0 değeri taşırken, en sağ konumu yüzde 100 değerini taşır. Her bir görüntü özelliği için bir adet kulp olduğundan dolayı, arayüzde on sekiz adet kulp vardır. Bu

kulpların başlangıç değerleri, hiçbir özellik seçili olmadığından dolayı 0 olarak belirlenmiştir. Bu durumda bütün kulplar sola yaslanmış halleriyle başlarlar. (Şekil 3.69)

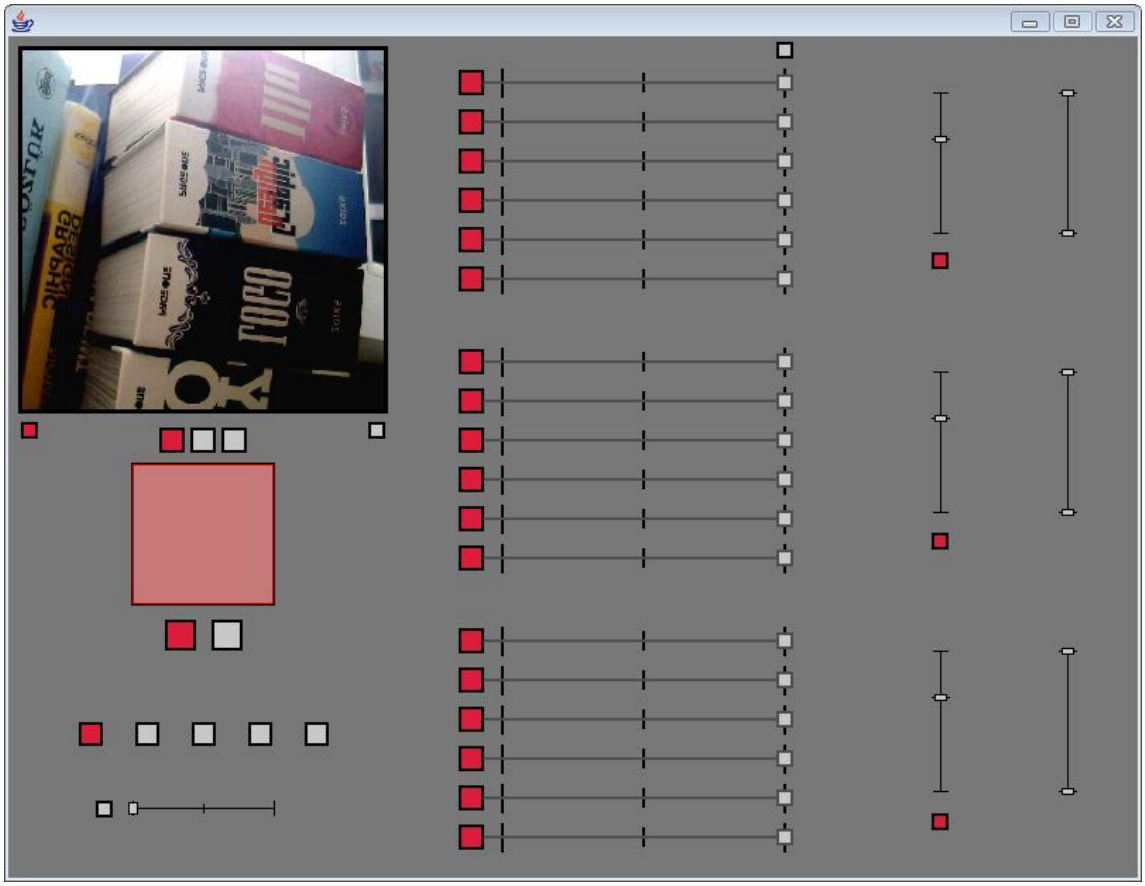


Şekil 3.69 Her bir görüntü özelliğine denk düşen oran belirleme kulpu

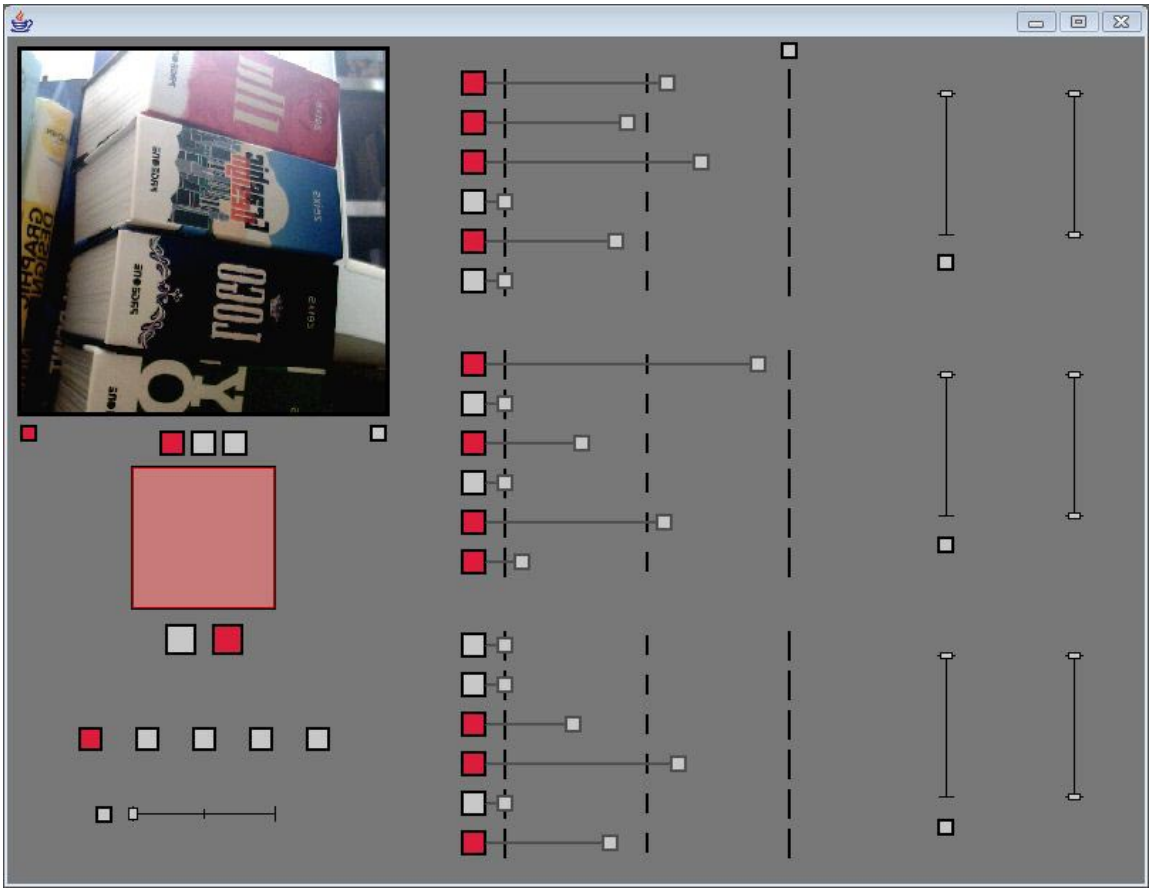
Bu kulplar yukarıdaki şekilde görüldüğü gibi, alt alta sıralanmış altışarlı gruplar halindedirler. Her altılı grup, bir ses özelliğine denk düşer. İlk altılı grup, ses frekans değeri için belirleyicidir. İkinci altılı grup, ses yüksekliği için belirleyicidir. Üçüncü altılı grup, sesin pan değeri için belirleyicidir. Kullanıcı her bir ses değeri için, bu kulplar aracılıyla kontroller sağlayarak istediği görüntü özelliklerini istediği oranda birleştirebilir.

Kullanıcı, görüntü özelliklerinin hepsinin yüzde yüz oranda ses değerlerine katılmasını isteyebileceği gibi, bu görüntü özelliklerini birbirlerinden bağımsız şekilde farklı oranlarda birleştirebilir. Kullanıcının kulplar ile ilgili olan etkileşimi sürekli devam

edeceğinden dolayı ve açık kapalı ilişkisinin olmaması nedeniyle bu kulpların dolgu renkleri hangi değeri alırlarsa alsınlar sürekli gri olarak kalır. (Şekil 3.70 ve Şekil 3.71)

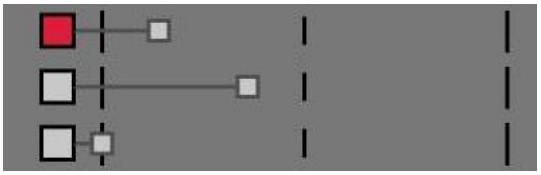


Şekil 3.70 Kullanıcının ses değerlerine aktarılacak olan görüntü özelliklerini yüzde yüz oranında birleştirmesi durumunda kulpların görünümü



Şekil 3.71 Kullanıcının ses değerlerine aktarılacak olan görüntü özelliklerini farklı oranlarda birleştirmesi durumunda kulpların görünümü

Burada, kulplar ile ilgili olan bir diğer özellik, kullanıcının her hangi bir görüntü özelliğinin ses kanalı değerlerine aktarılmasını engellemişken, yine de kulplar ile bu özelliğinin otomatik olarak yüzde sıfır değerini alması zorunluluğunun olmayışıdır. Bunun sebebi ise, işitsel-sanat icracısının, görüntü değerlerini belirlediği oranlarda, ara ara bu özelliği açıp kapayarak ses değerleri oluşturmak isteyebileceğidir. (Şekil 3.72)



Şekil 3.72 Görüntü özelliği kapalıyken oran belirleyen kulpun serbest olarak belirlenmesi

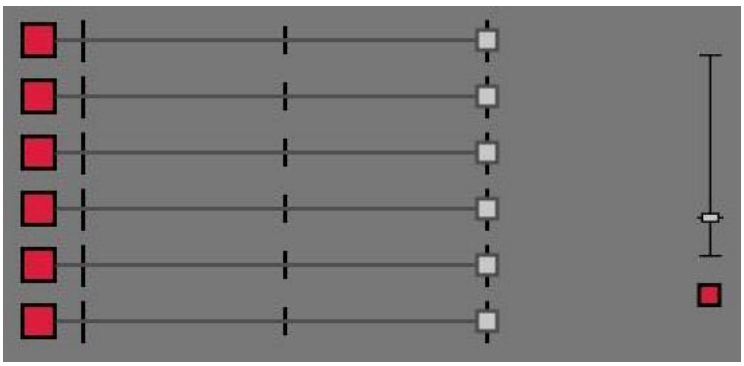
Bütün kulpların kaynak değerinin daha rahat görülmesi ve oranların daha açık kıyaslanabilmesi için, kulplar ile ilgili butonlar arasında gri renkli çizgiler çizilmiş durumdadır. Bu çizgiler, kullanıcının, kulpları görüntü değeri oranlarını belirlemek için sürüklemesi ile uzarlar ya da kısalırlar.

3.4.2.12 Ses sinyal değerlerinin gösterilmesi ve belirlenmesi

Aracın ses sinyal kanallarının üç özelliğinin (frekans, ses yüksekliği, pan), görüntü değerlerine bağlanabilme imkanı nedeniyle, değerlerinin gösterimi ve bunların kontrol edilmesinin gösterimi özel bir önem kazanmaktadır. Burada önemli olan, ses sinyal değerlerinin hem görüntü kaynaklı olarak dinamik değer alma özelliği hem de kullanıcı tarafından belirlenebilme özelliğinden dolayı sürgülü olması ve taşınabilir bir kulp olması gerekliliğidir. Böylece hem gerçek zamanlı olarak üretilen görüntü parametrelerine bağlı olarak güncellenebilecek, hem de kullanıcı tarafından rahatça değiştirilebilecektir. Bu yüzden, arayüzde üç adet ses sinyal değerini gösteren eleman vardır. Bunlar dikey olarak hizalanmışlardır. Üstte bulunan frekans değerini, ortada bulunan ses yükseklik değerini, altta bulunan da pan değerini temsil eder ve gösterir. (Şekil 3.73)

Şekil 3.73 Ses sinyal kanallarının değerlerini gösteren kulplar

Arayüzde bu ses değerlerini gösteren her bir kulp, kendisine denk düşen altı adet görüntü özelliğinin karşısında bulunur. Böylece kullanıcı, on sekiz adet görüntü özelliğini gösteren kulp ile bunlarla altılı ilişkilere giren ses sinyal kanalı değerlerini gösteren kulplar arasında rahatça ilişki kurabilecektir. (Şekil 3.74)



Şekil 3.74 Arayüzde ses sinyal kanalının değerini gösteren kulp ile buna karşılık gelen altı adet görüntü özelliği kulpunun konumları

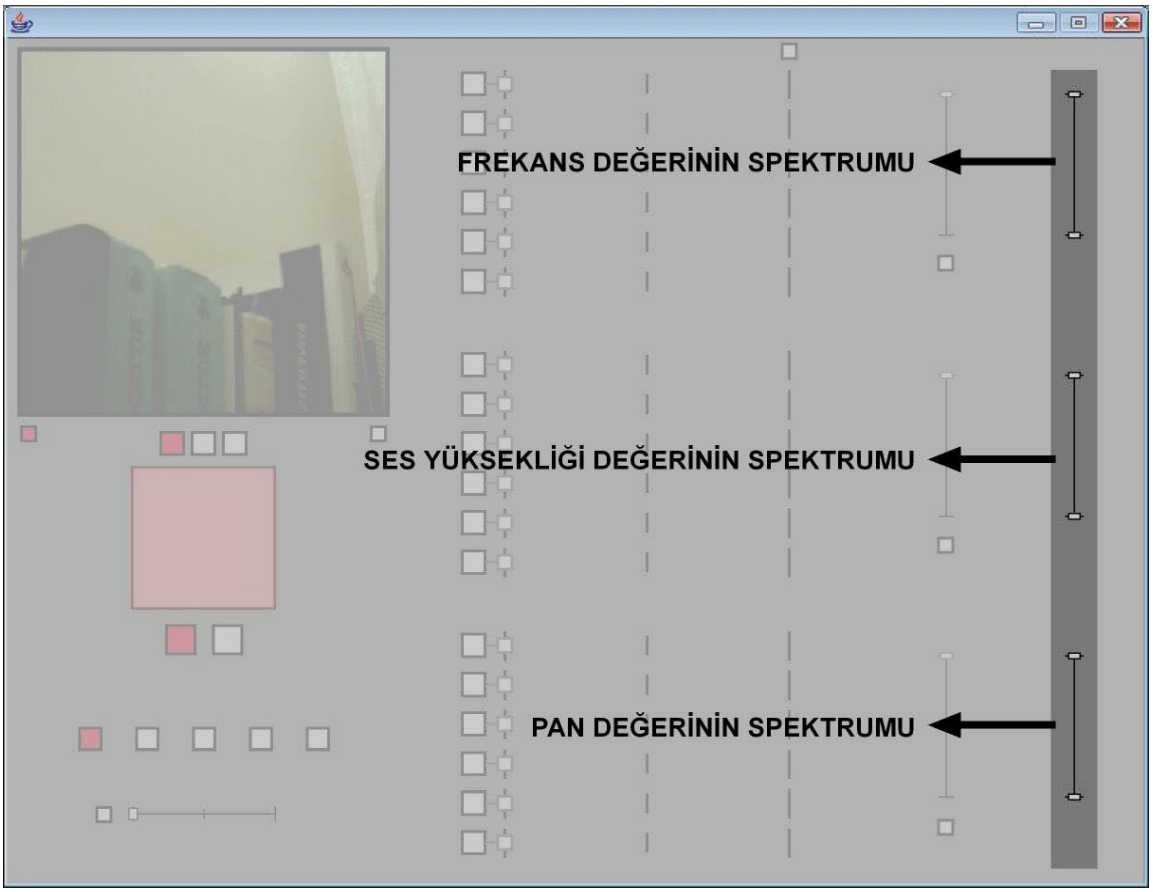
Bu kulplar, değerlerini yukarı-aşağı doğrultuda hareket ederek alırlar. En üst konumda iken minimum değerlerini, an alt konumda iken maksimum değerlerini alırlar. Başlangıç değerleri ise her üç kulp için 0'dır. Bunun nedeni, kullanıcının araç yeni çalışmaya başlamışken, hiçbir görüntü değerini ses sinyallerine bağlamış olamayacağıdır. (Şekil 3.75)

Şekil 3.75 Ses sinyal kanallarının başlangıç değerleri

3.4.2.13 Ses sinyal değerlerinin spektrum aralıklarının belirlenmesi

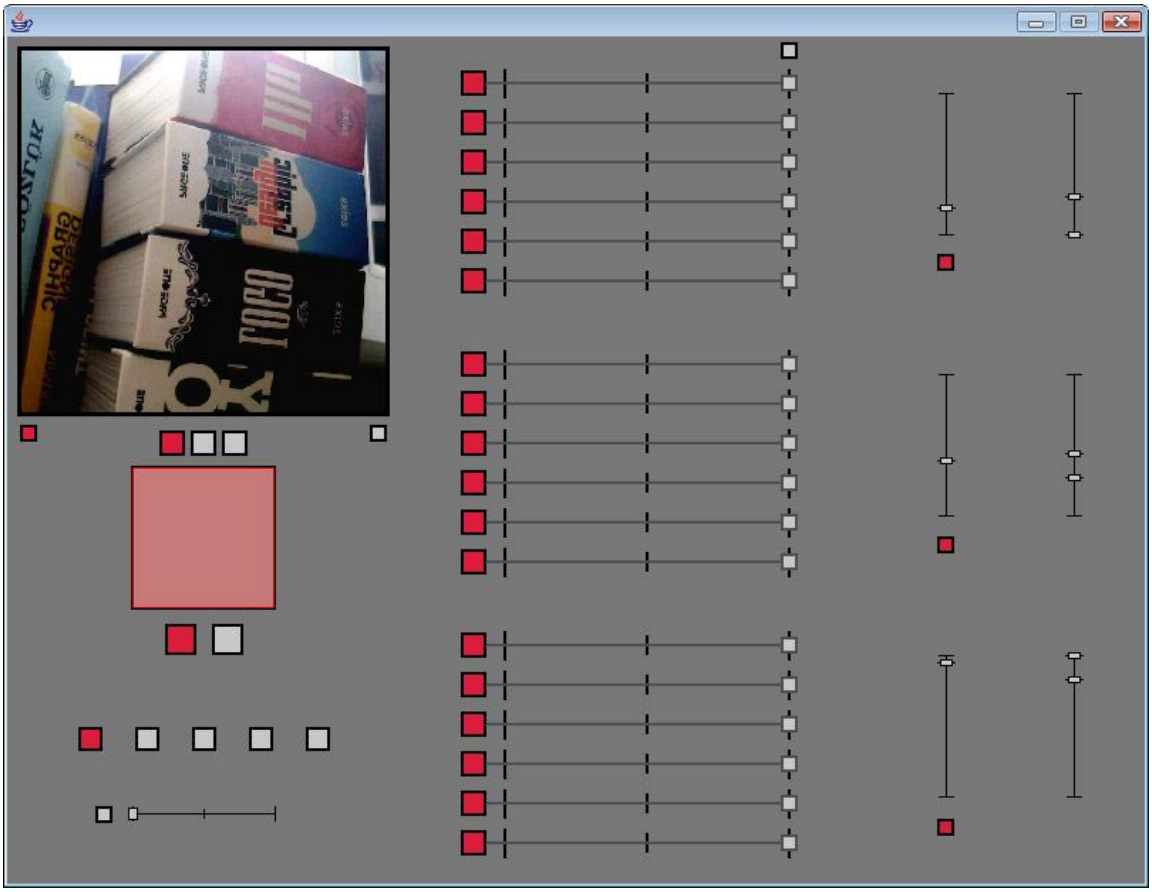
Aracın, ses sinyal değerleri ile ilgili olarak önemli bir özelliği spektrum aralıklarıdır. Kullanıcı, ses sinyal değerleri için istediği aralıkta nicelikler belirleyebilir. Bunun için, her bir ses sinyal değeri kulpu için bir adet spektrum belirleme alanı vardır. Arayüzde toplam üç adet ses sinyal kanalı gösteren kulp olduğu için bunlara karşılık olarak üç adet spektrum belirleme aracı vardır. Her bir spektrum aracında iki adet kulp bulunur. Bunlardan biri üst sınırı, diğeri de alt sınırı belirler. Bu kulplarda ses sinyal değeri

kulpları gibi, yukarı-aşağı doğrultuda hareket ederler. En üst noktaları minimum, en alt noktaları maksimum değer aralığını işaret eder. Başlangıç değerleri olarak, alt sınır kulpları minimum değerde, üst sınır kulpları ise maksimum noktasında bulunur. (Şekil 3.76)



Şekil 3.76 Spektrum kulplarının başlangıç konumları

Kullanıcı, spektrum aralıklarını istediği gibi belirleyerek, ses sinyal değerlerinin o aralıkta kalmasını sağlayabilir. Burada önemli bir nokta spektrum kulplarının dikey konumlarının birbirlerini aşamamasıdır. Minimum sınırı gösteren kulp, her zaman maksimum sınırı gösteren kulptan daha yukarıda bir konuma sahip olacaktır. Bu ilke sınırları içerisinde, kullanıcı istediği aralığı belirleyebilir. Ses sinyal değerleri, spektrum kulplarının aralığına göre kendisini güncelleyerek yeni nicelikler alır. (Şekil 3.77)



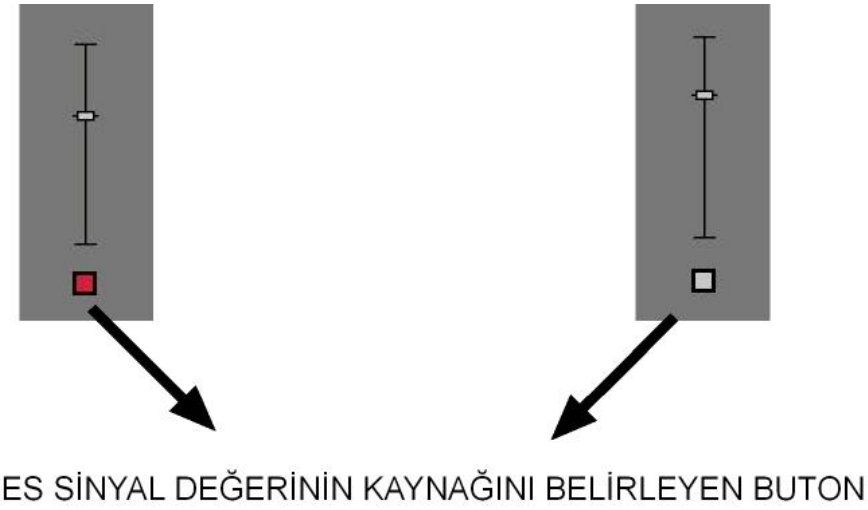
Şekil 3.77 Ses sinyal değerlerinin görüntü kaynağından gelen değerleri aynı oranda birleştirmesine rağmen spektrum aralıklarının nihai sonucu değiştirmesi

3.4.2.14 Ses sinyal değerlerinin kaynağının kamera ya da sabit değer olmasının belirlenmesi

Kullanıcı imkan olarak, ses sinyal kanallarının görüntü değerlerine bağlanabilen üç adet özelliğini, sabit değer vermek üzere el ile düzenleyebilir. Bunun için, öncelikle ilgili ses değerlerinin görüntü değerlerine bağlanıp bağlanmayacağına karar vermelidir. Bunun için her ses sinyal değeri gösteren kulpun altında bir adet buton bulunur. (Şekil 3.78)

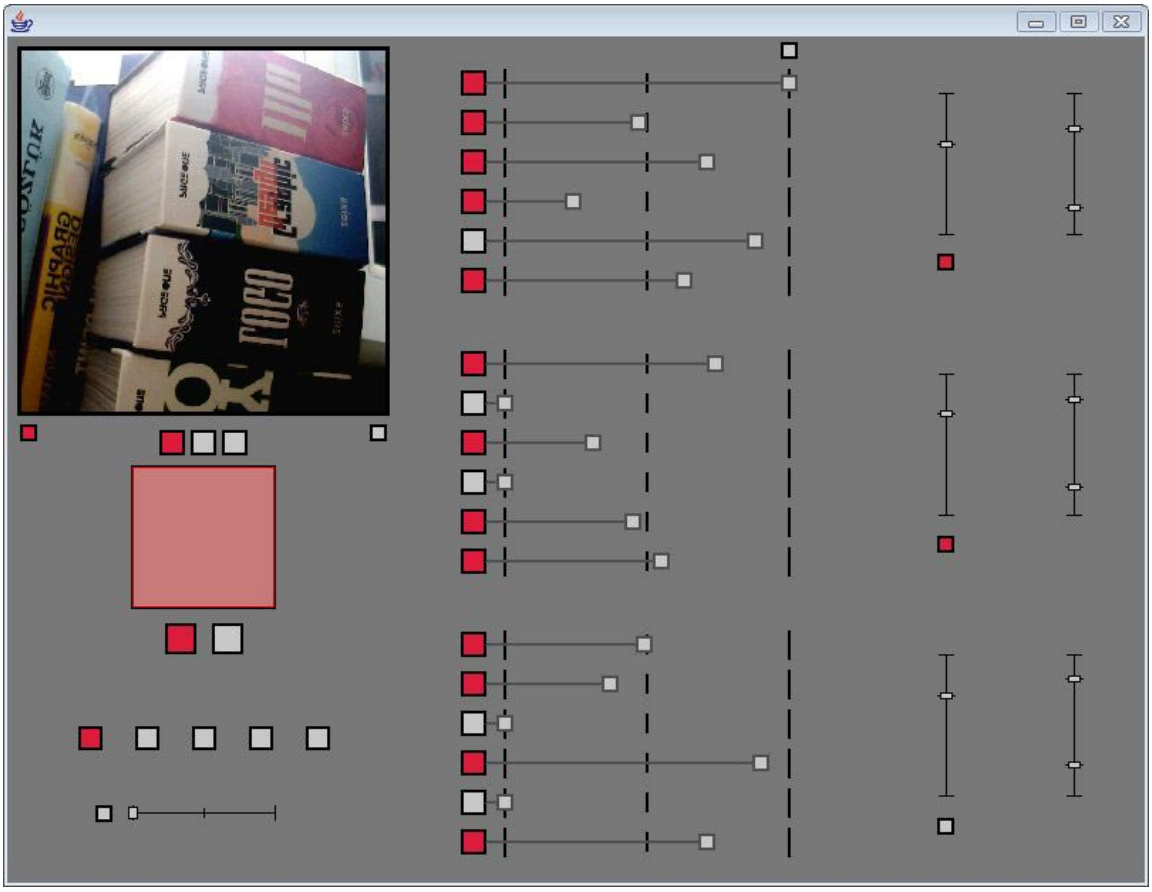
Şekil 3.78 Ses sinyal değerlerinin kaynağının belirlenmesini sağlayan butonların konumları

Bu buton seçilirse, görüntü değerleri ile ses sinyal değerleri arasında bağlantı kurulmuş olur. Bundan böyle ses sinyali için değerler el ile girilemez. Ancak eğer kullanıcı, bu butonu seçmedi ise, ses sinyal değeri el ile girilir ve görüntüden gelen herhangi bir değer bu ses kanalının değerlerini değiştirmez. Bu butonlar, seçili olma halleri için, dolgu rengi olarak kırmızıya boyanırlar. Seçili olmama hali için, dolgu rengi olarak griye boyanırlar. (Şekil 3.79)



Şekil 3.79 Ses sinyal değerlerinin kaynağını belirleyen butonların seçili olma ve seçili olmama görünümleri

Üç ses kanalı özelliği için olan bu butonlar, birbirlerinden bağımsız olarak seçilebilirler. Kullanıcı bir ya da iki ses kanalı özelliğini görüntü değerlerine bağlarken, bir diğerine el ile sabit değer girmek isteyebilir. Burada önemli bir nokta, ses sinyal değerlerinin elle belirlenmesine karar verilmiş ise de, görüntü için ayarlanmış olan değerler korunmaya devam eder. Bunun sebebi, işitsel-sanat icracısının, görüntü değerlerini el ile belirlediği süreçte, ara ara bu özelliği açıp kapayarak yeni ses değerleri oluşturmak isteyebileceğidir. (Şekil 3.80)



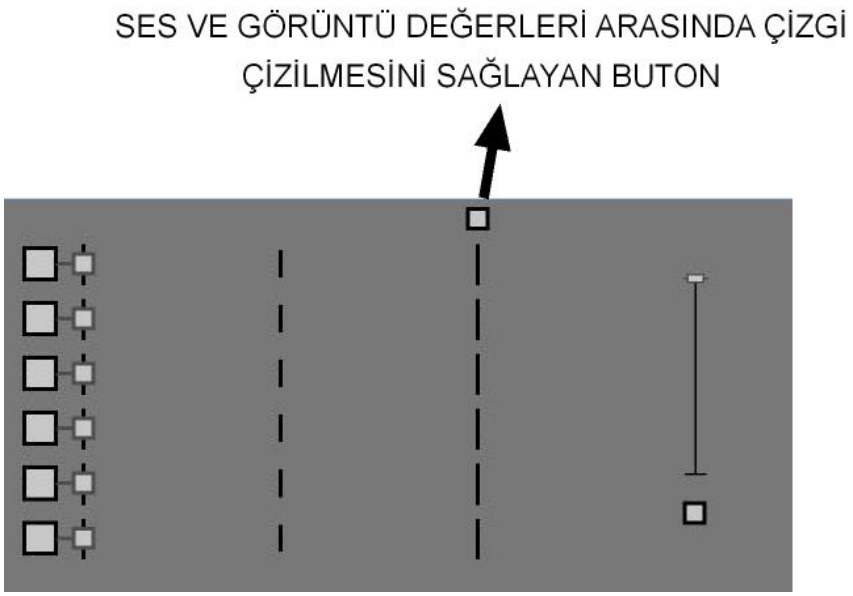
Şekil 3.80 Ses sinyal özelliklerinin görüntü değerlerine bağlanması ya da elle belirlenmesinin birbirinden bağımsız gerçekleşmesi

Bu butonlar başlangıç değeri olarak, ses sinyallerine elle sabit değer girilmesine izin veren konumda olurlar. Bu yüzden, kullanıcı kameraya bağlamak istediği bir özellik için, bu butonları seçerek aktif hale getirmelidir. Bu butonlar, başlangıçta gri dolgu rengiyle seçili olmama konumundadırlar.

3.4.2.15 Ses sinyal değerleri ile görüntü değerleri arasındaki çizgilerin gösterilmesi

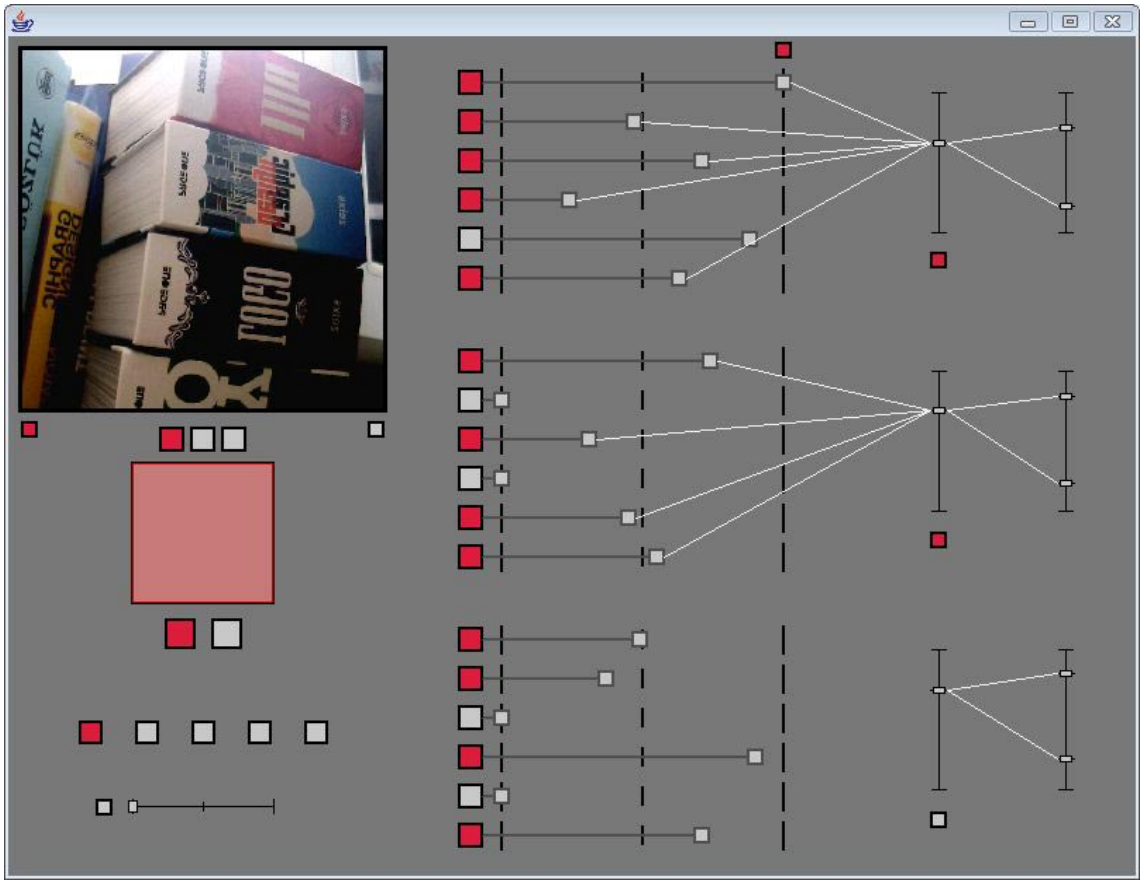
Aracın arayüzünün bir diğer özelliği, kullanıcının ses sinyal değerleri ile görüntü değerleri arasındaki ilişkileri daha rahat görebilmesi için, bu özellikleri temsil eden kulplar arasında çizgi çizilmesi seçeneğidir. Bu seçeneğin çalıştırıldığı buton, işlevi ile paralel olarak ses ve görüntü değerlerini gösteren alanın ortasında bulunur. Bu buton başlangıç değeri olarak seçili olmama konumundadır. Bu yüzden gri renk dolgusu ile

birlikte durur. Eğer kullanıcı bu seçeneği aktif hale getirirse, kırmızı renk dolgusuna döner. (Şekil 3.81)



Şekil 3.81 Ses ve görüntü değerleri arasında çizgi çizilmesini sağlayan butonun başlangıç değeri ve görünümü

Eğer kullanıcı bu seçeneği aktif hale getirirse, ses sinyal değerini gösteren üç adet kulp ile bunların karşısında bulunan spektruma ait iki adet kulp arasında ve yine ses sinyal değerini gösteren kulp ile eğer görüntü değerlerinin kameradan alınma butonu açıksa ilgili görüntü özelliklerine denk düşen kulplar arasında çizgi çizilir. Eğer kullanıcı el ile sabit değer girme seçeneğini seçtiyse, ses sinyal değerleri ile görüntü değerleri arasında çizgiler çizilmez. Bu durumda sadece ses sinyal değerini gösteren kulplar ile spektrum kulpları arasında çizgiler çizilir. (Şekil 3.82)



Şekil 3.82 Ses sinyal değerleri ile görüntü değerleri arasında ve ses sinyal değerleri ile spektrum kulpları arasında çizilen çizgiler ve ilgili butonun görünümü

Bu çizgiler, gerçek zamanlı olarak, her yeni değerde güncellenmektedir. Bu durum arayüzde hareketli çizgilerin oluşmasına imkan vermektedir. Böylece kullanıcı, bu hareketli çizgiler ile ses ve görüntü değerleri arasında bulunan ilişkileri daha rahat görebilmektedir.

4. SONUÇ ve YORUMLAR

Tez çalışması süresince kamera kaynaklı işitsel-sanat aracı ve arayüzünün teorik ve pratik temellerinin oluşturulmasına çalışıldı. Öncelikle işitsel-sanat alanının özellikleri ortaya konuldu. Ses ile görüntü ortamlarının ilişkisi ve etkileşimi üzerine gerçekleştirilmiş olan ve tezimizin arka planını oluşturan çalışmalar incelendi. Bunlardan çıkarılan sonuçlar ile kendi aracımızın taşıması gereken temel nitelikler belirlendi. Ardından, belirlediğimiz hedefe ulaşabilmek için kullanacağımız malzemeler ve bunları değerlendireceğimiz araç geliştirme ortamları incelendi ve yöntemler oluşturuldu. Son olarak belirlenen amaca uygun etkileşimli bir işitsel-sanat aracı ve arayüzünün sistemi tasarlandı ve kodlamaları yapılarak çalışır hale getirildi.

Bu tez çalışmasının nihai amacı, işitsel-sanatın temel öğesi olan ses nesnelerinin kaynağını kamera görüntülerinden oluşturarak, kamerayı bir ses enstrümanına dönüştürmektir. Ses kamera ile belirlenerek, piksellerin frekanslara dönüştürülmesi sağlandı. Etkileşim tasarımı kararlarımızın gerçekleştirilmesi ve sistemin sahip olduğu veri modelleri ile işlevsel mimarisinin bilgi teknolojileri açısından uygulanabilirliği gösterildi.

Kamera kaynaklı işitsel-sanat aracının konseptinin belirlenmesi ve uygulamasının gerçekleştirilmesine rağmen, tarafımızdan bu tasarımın geliştirilmesinin sonlandığı iddia edilmemektedir. Özellikle aracın çalışmasını sağlayan algoritmalar ve kodlar, gelinebilecek en yüksek performansa sahip değildir. Bunun gerçekleştirilmesi için, genişletilmiş bir yazılım geliştirme ekibinin olması gerekmektedir.

Bunun yanında, çalışma sadece bir kamera girişi ile gerçekleştirilmiştir. Bundan sonra yapacağımız çalışmalar ile birden fazla kameranın kullanıcının (sound-artist) kontrolüne verilmesi sağlanacaktır. Bu durumda, farklı görüntülerin üst üste gelmesi ile kullanıcı, seslerin kaynağını oluşturan görsel yapının da inşasına yönelecektir. Böylece oluşturulan ses sinyallerinin yelpazesi, görüntü ortamında oluşan ilişkilerin zenginliğinin artması ile daha da genişleyecektir. Farklı parçaların yan yana gelerek

görsel bir bütünü oluşturması, etkileşim tasarımı olarak farklı ses sinyal parçalarının yan yana gelerek birbirlerini etkilemelerinin yolunu açacaktır.

Tez çalışmamızda, kamera kaynaklı işitsel-sanat aracının etkileşimlerinin ve düzeneğinin tasarlanarak uygulaması yapılmıştır. Bu noktadan sonra, grafik arayüz tasarımı için kavramsal bir çerçevenin çizilmesi ve bunun uygulanması, insan bilgisayar etkileşimi alanına yönelik olarak kullanıcı davranışları ile ilgili saptamaların yapılması, makine-sistem öğrenilebilirliğinin tartışılması ve bu araç ile üretilen performansların estetik olarak analiz edilerek sınıflandırılması daha sonraki akademik araştırmalarımızın konularını oluşturacaktır.

KAYNAKÇA

BEHRMAN, D., 1991, Designing Interactive Computer-Based Music Installations, *Contemporary Music Review*, 1991, Vol. 6, Part 1, 139-142

BELL, B., KLEBAN, J., OVERHOLT, D. VE PUTNAM, L., THOMPSON, J. ve KUCHERA-MORIN, J., 2007, The Multimodal Music Stand, *Proceedings of the 2007 Conference on New Interfaces for Musical Expression (NIME07)*, June 7-9, 2007, New York, NY, USA, 62-65

BIRCHFIELD, D., PHILLIPS, K., KIDANÉ, A. ve LORIG, D., 2006, Interactive Public Sound Art: a case study, *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06),* June 4-8, 2006, Paris, France, 43-48

BORCHERS, P., HADJAKOS, A. ve MUHLHÄUSER, M., 2006, MICON A Music Stand for Interactive Conducting, *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06),* June 4-8, 2006, Paris, France, 254-259

CAMURRI, A., COLETTA, P., DRIOLI, C., MASSARI A. ve VOLPE G., 2005, Audio Processing in a Multimodal Framework, *Audio Engineering Society Convention Paper,* Presented at the 118th Convention, May 28–31, 2005, Barcelona, Spain

DANNENBERG, R., B., 2002, A Language for Interactive Audio Applications, *Proceedings of the 2002 International Computer Music Conference*, 1996, San Francisco: International Computer Music Association, 509-515

DANNENBERG, R., B. ve BRANDT, E., 1996, A Flexible Real-Time Software Synthesis System, *Proceedings of the 1996 International Computer Music Conference*, San Francisco: International Computer Music Association, August, 1996, 270-273

DEWITT, T., 1987, Visual Music: Searching for an Aesthetic, *Leonardo*, 1987, Vol. 20, No. 2, 15-122

GAYFORD, M., L., 1970, *Electroacoustics: Microphones, Earphones and Loudspeakers*, Newnes-Butterworth, London, 0-408-00026-0

HAWKSFORD, M., O., 1991, An introduction to Digital Audio, *Proceedings of the l0th International AES Conference*, September, 1991, London, T3-T42

KANDINSKY, W., 2005, *Sanatta Ruhsallık Üzerine*, 2.Baskı, Gülin Ekinci(çev.), Altıkırkbeş Yayın, İstanbul, 975-8467-40-9 (orijinal yayın: 1911)

KARAHOCA, D., KARAHOCA, A., 1998, *Yönetim Bilişim Sistemleri ve Uygulamaları*, 1.Baskı, Beta Yayınevi, İstanbul, 975-486-728-3

KILIÇ, L., 1994, *Görüntü Estetiği*, 1.Baskı, Yapı Kredi Yayınları, İstanbul, 975-363-332-7

KUŞÇU, H. ve AKGÜN, B., T., 2005, A New Approach to Interactive Performance Systems, *Proceedings of the 13th annual ACM international conference on Multimedia*, November 6-11, 2005, Singapore, 578-581

LEVIN, G., 2000, Painterly Interfaces for Audiovisual Performance, *Master of Science,* Massachusetts: Massachusetts Institute of Technology, Media Arts and Sciences

LEVIN, G. ve LIEBERMAN, Z., 2004, In-Situ Speech Visualization in Real-Time Interactive Installation and Performance, *Proceedings of The 3rd International Symposium on Non-Photorealistic Animation and Rendering*, June 7-9, 2004, Annecy, France, 7-14

LEVIN, G. ve LIEBERMAN, Z., 2005, Sounds from Shapes: Audiovisual Performance with Hand Silhouette Contours in The Manual Input Sessions, *Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05)*, May 26-28, 2005, Vancouver, BC, Canada, 115-120

MEADOWS, S. ve AKLEMAN, E., 2000, Abstract Digital Paintings Created with Painting Camera Technique, *Proceedings of D'ART 2000 / Information Visualization*, July, 2000, London, United Kingdom

PUCKETTE, M., 1996, Pure Data: Another Integrated Computer Music Environment, *Proceedings of the International Computer Music Conference*, 1996, San Francisco: International Computer Music Association, 269-272

RAMAKRISHNAN, C., FREEMAN J. ve VARNIK, K., 2004, The Architecture of Auracle: a RealTime,Distributed, Collaborative Instrument, *Proceedings of the 2004 Conference on New Interfaces for Musical Expression (NIME04)*, June 3-5, 2004, Hamamatsu, Japan, 100-103

SANKUR, B., 2004, *İngilizce-Türkçe Ansiklopedik Bilişim Sözlüğü*, 2.Baskı, Pusula Yayıncılık, İstanbul, 975-6477-03-2

SMITH, J., AKLEMAN, E., DAVISION, D. ve KEYSER, J., 2004, MultiCam: A system for Interactive Rendering of Abstract Digital Images, *Proceedings of the Bridges: Mathematical Connections in Art, Music and Science*, August, 2004, Winfield, Kansas, USA, 265-272

SÖZEN, M., 2003, *Sinemada Ses Kullanımı*, 1.Baskı, Detay Yayıncılık, Ankara, 975-8326-55-4

ŞENOVA, B., MURATOĞLU, E. ve ERKAL, E., 2007, Sound-Art İşitsel-Sanat ve Zaman İçindeki Gelişimi, *Bant Dergisi*, 35, 93-95

ZEREN, M., A., 1995, *Müzik Fiziği*, 1.Baskı, Pan Yayıncılık, İstanbul, 975-7652-46-6

İnternet Kaynakları

1. *Sound art*, Wikipedia the free encyclopedia, http://en.wikipedia.org/wiki/Sound_art, [erişim tarihi: 05, 08, 2007]

2. NEUHAUS, M., *Sound art*?, sergi açılış konuşması "Volume: Bed of Sound", P.S.1 Contemporary Art Center, New York, July 2000, http://www.max-neuhaus.info/soundworks/soundart, [erişim tarihi: 01, 07, 2007]

3. *List of sound artists*, Wikipedia the free encyclopedia, http://en.wikipedia.org/wiki/List_of_sound_artists, [erişim tarihi: 05, 08, 2007]

4. *Pictures at an Exhibition*, Wikipedia the free encyclopedia, http://en.wikipedia.org/wiki/Pictures_at_an_Exhibition, [erişim tarihi: 27, 06, 2007]

5. *Theremin*, Wikipedia the free encyclopedia, http://en.wikipedia.org/wiki/Theremin, [erişim tarihi: 07, 05, 2007]

6. *An Audiovisual Environment Suite,* http://acg.media.mit.edu/people/golan/aves/, [erişim tarihi: 12, 02, 2007]

7. *Dialtones (A Telesymphony) Final Report*, http://www.flong.com/projects/telesymphony/, [erişim tarihi: 28, 06, 2007]

8. *Messa Di Voce*, http://www.tmema.org/messa/messa.html, [erişim tarihi: 17, 07, 2007]

9. *Max/MSP*, http://www.cycling74.com/products/maxmsp, [erişim tarihi: 12, 04, 2007]

10. *Pure Data*, http://puredata.info/, [erişim tarihi: 16, 04, 2007]

11. ŞENGÜL, T., *Müzik Fiziği*, http://www.bgst.org/muzik/egitim/muzikfizigi.html, [erişim tarihi: 23, 08, 2007]

12. *Waveform*, http://www.search.com/reference/Waveform, [erişim tarihi: 12, 08, 2007]

13. *Munsell color system*, Wikipedia the free encyclopedia, http://en.wikipedia.org/wiki/Munsell_color_system, [erişim tarihi: 27, 07, 2007]

14. *Hardware Requirements*, http://eyecon.palindrome.de/, [erişim tarihi: 18, 06, 2007]

15. *Developer Resources for Java Technology*, http://java.sun.com/, [erişim tarihi: 26, 07, 2007]

16. *Processing 1.0 (Beta)*, http://www.processing.org/, [erişim tarihi: 08, 06, 2007]

17. *Video \ Libraries \ Processing 1.0 (Beta),* http://www.processing.org/reference/libraries/video/index.html, [erişim tarihi: 08, 06, 2007]

18. *Minim,* http://code.compartmental.net/tools/minim/, [erişim tarihi: 12, 06, 2007]

ÖZGEÇMİŞ

Doğum tarihi	: 05.08.1980
Doğum yeri	: İstanbul
Lise	: 1994 - 1998 Şişli Çağlayan Süper Lisesi
Lisans	: 1989 - 2004 Yıldız Teknik Üniversitesi, Sanat ve Tasarım Fakültesi, İletişim Tasarımı Bölümü
Yüksek Lisans	: 2005 – 2008 Bahçeşehir Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Bilgi Teknolojileri Yüksek Lisans Programı

Çalıştığı kurum:

2005 – 2008	: Bahçeşehir Üniversitesi, İletişim Fakültesi, Görsel Sanatlar ve Görsel İletişim Tasarımı Bölümü - Araştırma Görevlisi