

Yıldız Teknik Üniversitesi
Sanat ve Tasarım Fakültesi
Müzik ve Sahne Sanatları Bölümü
Duysal Tasarım Programı

Bitirme Projesi

Çizim Tabletleri İçin Fiziksel Modelleme Temelli Sentezleyici

Proje Danışmanı:

M. Kemal Karaosmanoğlu

Projeyi Hazırlayan:

03083002 Utku Uzmen

İstanbul, 2008

İÇİNDEKİLER

Şekil Listesi.....	iv
Tablo Listesi.....	v
Önsöz.....	vi
Özet.....	vii
Abstract.....	viii
1. Giriş.....	1
1.1 Sentezleyici Tasarımı	2
1.2 Arayüz Tasarımı.....	2
1.3 Sentezleyici-Arayüz Eşleşmesi	2
1.4 Arayüzden Sentezleyiciye Doğru Tasarım	4
2. Arayüz ve Sentezleme Tekniği Seçimi.....	5
2.1 İnsan-Bilgisayar Arayüzü Olarak Çizim Tabletleri	5
2.1.1 Ergonomik Özellikler	6
2.1.2 Ses Uygulamalarında Kullanım	6
2.1.3 Ses Uygulamalarında Diğer Kontrol Aygıtlarıyla Karşılaştırma	7
2.2 İncelenen Akustik Enstrümanlar	8
2.2.1 Vibrafon.....	9
2.2.2 Armonika	10
2.2.3 Tibet Çanağı.....	12
2.3 Sentezleme Tekniği Olarak Fiziksel Modelleme	13
2.3.1 Çizim Tabletlerinin Mekanik Özellikleri İçin Temel Model.....	14
2.3.2 Temel Model İçin Algoritma Seçimi.....	14
3. Teori ve Algoritma.....	16
3.1 Sürtünme	16
3.2 Öz Titreşim Modları.....	18
3.3 “Banded Waveguide” Temelli Algoritma	21
4. Uygulama	26
4.1 Altyapı	26
4.1.1 Ses Donanımına Erişim	27
4.1.2 Çizim Tabletiyle İletişim	28
4.1.3 Ses Dizileri İçin Scala Dizi Formatı.....	28
4.2 Çekirdek Nesne Sınıfları.....	29
4.2.1 Filtre Sınıfı	29
4.2.2 Delay Sınıfları.....	30

4.2.3	Waveguide Sınıfı.....	31
4.3	Ses Çıkışı	35
4.4	Yazılım Arayüzü	36
4.4.1	Araç Çubuğu	36
4.4.2	Parametre Paneli.....	36
4.4.3	Dizi İçeriği Paneli ve Durum Çubuğu	39
4.4.4	İcra Alanı	39
4.5	Performans İyileştirmeleri	41
5.	Sonuçlar	43
	Kaynaklar	45
	Ek-1: Yazılım Kaynak Kodu.....	49

Şekil Listesi

Şekil 2.1 Vibrafon (14)	9
Şekil 2.2 Armonika (16)	10
Şekil 2.3 Benjamin Franklin'in Armonika tasarımı (17)	11
Şekil 2.4 Tibet çanağı ve tokmağı (<i>puja</i>) (19)	12
Şekil 3.1 Sembolik <i>stick-slip</i> sürtünme mekanizması	16
Şekil 3.2 Metal plaka üzerinde kum tanelerinin oluşturduğu Chladni motifleri (29)	18
Şekil 3.3 Titreşim halindeki bir çanın hologramı (30)	19
Şekil 3.4 Temel <i>waveguide</i> akış şeması	22
Şekil 3.5 Bileşenlerin toplandığı (<i>commuted</i>) temel <i>waveguide</i> şeması	22
Şekil 3.6 Temel <i>banded waveguide</i> akış şeması	23
Şekil 3.7 Tasarlanan <i>banded waveguide</i> modelinin akış şeması.....	24
Şekil 4.1 Titreşim modlarının ağırlık fonksiyonları ve spektral karakterleri	38
Şekil 4.2 Yazılım arayüzünde temsili çubuklar	40

Tablo Listesi

Tablo 3.1 İki ucu serbest çubuk için titreşim modlarının frekans oranları	21
Tablo 3.2 Bir ucu sabitlenmiş çubuk için titreşim modlarının frekans oranları	21
Tablo 3.3 İki ucu menteşe destekli çubuk için titreşim modlarının frekans oranları	21
Tablo 4.1 "Armonika.vmod" dosyası için titreşim modlarının frekans oranları	38
Tablo 4.2 "Tibet Çanağı.vmod" dosyası için titreşim modlarının frekans oranları	38
Tablo 4.3 "Akordlu Çubuk.vmod" dosyası için titreşim modlarının frekans oranları	39
Tablo 4.4 "Teneke Kutu.vmod" dosyası için titreşim modlarının frekans oranları	39

Önsöz

Çoğu zaman teknoloji alanındaki gelişmelerin çıkış noktası, yapılagelen işlerin daha kolay, daha çabuk veya daha verimli şekilde yapılır hale getirilmesi ihtiyacıdır. Fakat bu gelişmelerden elde edilen sonuçları sadece bu yönde kullanmak yeterli olmayabilir. Yeni teknolojilerin potansiyellerinin tamamından yararlanmak için, onlara özel yöntemlerin, yaklaşımların ve kullanım alanlarının bulunması gerekir.

Müzik teknolojisi alanında bu durumun birçok örneği bulunabilir. Bunlardan biri, Roland firmasının 1980’lerde ürettiği, ‘demo’ kayıtlarında davul setinin (ve davulcunun) yerini tutacak bir cihaz olarak piyasaya sürülmüş, fakat daha sonra elektronik dans müzik türlerindeki kullanımıyla asıl kimliğini kazanmış TR-909 davul makinasıdır.

Bu çalışmada, müzik teknolojisi alanında yazılım ve donanım tasarımında, var olan birikimlerden yararlanırken, tasarlanan yeniliklerin önünü kapatabilecek ‘gelenek’lerden kaçınmak gerektiği ve her tasarımın yeni bir yaklaşıma ihtiyaç duyabileceği vurgulanmak istenmiştir.

Bu çalışmanın ortaya çıkmasında, fikirleri ve yazılım test aşamasındaki yardımları için Berke Beyazay’a ve danışmanım M. Kemal Karaosmanoğlu’na teşekkür ederim.

Özet

Sentezleyicilerin ve kontrol arayüzlerinin birbirinden bağımsız hale gelmesi, bu iki unsur arasında bir soyutlamanın oluşmasına yol açmıştır. Bu soyutlama, bir yandan sentezleyicilerin ve kontrol arayüzlerinin çeşitli yönlerde gelişmesinin önünü açarken, diğer yandan her iki unsurun da kullanım potansiyellerinin tamamen ortaya çıkmasında engel oluşturmaktadır. Bu durum özellikle fiziksel modelleme teknikleri için geçerlilik kazanmıştır.

Fiziksel modelleme algoritmalarının eş zamanlı kullanımlarının mümkün hale gelmesiyle, bu algoritmaların etkin biçimde kullanılabilmesi için, modellerle ileri düzeyde eşleşen kontrol mekanizmalarının tasarımı da önem kazanmıştır.

Bu çalışmada amaçlanan, sentezleyici ile kontrol arayüzü eşleşmesinde, iki unsurun birbirine yönelik tasarlanmasının, kullanılabilirlik ve ifade olanağı açısından yarattığı sonuçların somut bir şekilde gösterilmesidir. Bu yönde, var olan bir kontrol mekanizmasından yola çıkılmış, bu kontrol mekanizmasına uygun bir model tasarlanarak canlı icraya yönelik bir uygulama geliştirilmiştir. Seçilen kontrol arayüzü, görsel tasarım alanında kullanılan çizim tabletleri; kullanılan sentezleme tekniği, tekbiçim ince çubukları modelleyen bir *banded waveguide* algoritmasıdır.

Uzmen, Utku. *Çizim Tabletleri İçin Fiziksel Modelleme Temelli Sentezleyici.* İstanbul: Yıldız Teknik Üniversitesi, Sanat ve Tasarım Fakültesi, Müzik ve Sahne Sanatları Bölümü, Duysal Tasarım Programı Lisans Bitirme Tezi, 2008.

Abstract

As synthesizers and control interfaces have grown to be independent elements, the use of an abstraction layer between them has become necessary to make complete electronic musical instruments. While this abstraction has made it possible for both synthesizers and control interfaces to advance in a multitude of directions, it can also prevent them from reaching their full potential by generalizing them to the point of exposing no distinct features to each other. This especially applies to physical modeling synthesis techniques.

With the application of physical modeling techniques as real-time algorithms in the recent years, the design of tightly coupled physical models and control interfaces has been found to be crucial for the effective and expressive use of these algorithms.

The purpose of this work is to demonstrate the results of such a tightly coupled design in terms of usability and expressiveness. With this purpose, an existing control interface was chosen, and a suitable algorithm was designed and implemented in a computer application geared towards live performance. The selected control interface is the graphics tablet, and the algorithm is a banded waveguide model of uniform slender bars.

Uzmen, Utku. *A Physical Modeling Synthesizer for Graphics Tablets*. Istanbul: Yıldız Technical University, Faculty of Art and Design, Department of Music and Performing Arts, Audio Design Program Undergraduate Thesis, 2008.

1. GİRİŞ

20. yüzyılın başlarında, elektronik ses üretme ve işleme yöntemlerinin yeni tını arayışlarına bir cevap oluşturacak noktaya gelmesiyle, bu sistemleri kontrol etme yaklaşımları da bir araştırma ve deney alanı olarak önem kazandı. Tasarlanan elektronik aygıtların özgün nitelikleri ve getirdikleri yeni olanaklar, kontrol mekanizmalarıyla ses üretme ve işleme yöntemlerinin birbirlerine bağlı biçimde evrilmelerini kaçınılmaz kıldı. Bu paralellik, Theremin'in dokunmadan uygulanan kesintisiz jestlere dayalı icrasında, ondes Martenot'nun vibrato için hareket payı olan klavyesi ve aralıksız ton değişimi sağlayan 'metal halka' mekanizmasında ve elektronik *sackbut*'un sol el için tını şekillendirme kontrollerinde görülebilir. Fakat dış kaynaklı elektrik sinyalleriyle kontrol edilebilen modüler sentezleyicilerin ortaya çıkmasıyla birlikte, sentezleyici ve kontrol mekanizması birbirlerinden bağımsız aygıtlara dönüştüler (1).

Bu ayrılma, 'kurdele' mekanizmalarından beyin dalgaları kullanan kontrol aygıtlarına kadar çok çeşitli kontrol yaklaşımlarının tasarlanmasını destekledi. 1980'lerin başında MIDI standardının yaygınlaşmasıyla, dijital temelli aygıtlar arasında da aynı bağımsızlığın önü açılmış oldu. Bu bağımsızlığın sözü edilen avantajına karşılık dezavantajı, birbirlerinin özelliklerinden ancak yüzeysel olarak haberdar olan sentezleyici-kontrol aygıtı eşleşmelerinin oluşmasıydı. Piyano klavyesini temel alan bir klavye sistemi ve diğer sentezleyici özelliklerini genellemelerle temsil eden ikincil düğmeler, tekerlekler ve benzeri bileşenlerden oluşan arayüzler, baskın kontrol aygıtlarının temeli oldu.

Sentezleme yöntemleri de, elektronik müzik enstrümanlarının popüler kültür içinde yaygın olarak kabul görmesine paralel bir hızla evrimlerine devam etti. Güncel teknolojinin getirdiği olanaklar, bugünün elektronik müzik bestecisine, analog sentezleme yöntemlerinden örnekleme bazlı algoritmalar ve fiziksel modelleme sistemlerine kadar çok sayıda ses üretme tekniği sunmakta.

1.1 Sentezleyici Tasarımı

Dijital temelli teknolojilerin olgunlaşması, özellikle son 10 yılda üretilen sentezleyicilerin büyük çoğunluğunun dijital sentezleyici sınıfında olmasını mümkün hale getirmiştir. Bu sentezleyiciler form faktörleri itibariyle iki ana gruba ayrılabilirler: Kendi donanımları üzerinde çalışan sentezleyiciler ve bilgisayar yazılımı olarak kullanılan (diğer bir deyişle form faktörü olmayan) sentezleyiciler.

Kendi donanımları üzerinde çalışan sentezleyicilerde, sentezleme akışının yönlendirilmeye açık parametreleri için donanım üzerinde bulunan fiziksel mekanizmalara ek olarak, söz konusu parametreler MIDI üzerinden de kontrole açılır. Yazılım formundaki sentezleyiciler de çoğunlukla bilgisayar ekranındaki grafik arayüzlerine ek olarak bütün parametrelerini MIDI üzerinden dışarıdan müdahaleye açarlar.

1.2 Arayüz Tasarımı

Elektronik müzik enstrümanları ve ilgili aygıtların giderek büyüyen bir pazar payı oluşturmasının sonucu olarak, son 10 yıl içerisinde üretilen kontrol aygıtları ve tasarlanan arayüz çeşitlerinin sayısı önemli bir artış göstermiştir. Bu ürünler arasında, geleneksel piyano klavyesini veya standartlaşmış düğme ve tekerlekleri temel alan arayüzlerin yanı sıra, dokunmaya duyarlı ekranlar (2), uzaktan hareket algılayıcıları (3) ve biyolojik sinyal çeviricileri (4) gibi var olan donanımsal teknolojinin uç noktalarından yola çıkan, daha küçük hedef kitleleri olan aygıtlar bulunmaktadır.

Güncel kontrol aygıtlarının büyük çoğunluğu, kontrol bileşenlerinden elde ettikleri değerleri, kontrol edilen hedefe MIDI mesajları olarak gönderirler. MIDI standardının soyut kontrol mesajı tanımlamasına bağlı olarak, kontrol aygıtı nasıl bir parametre için mesaj gönderdiğini bilmeyebilir.

1.3 Sentezleyici-Arayüz Eşleşmesi

MIDI standardı, buluşturduğu aygıtlarla ilgili sadece yüzeysel tanımlamalar yapmaktadır. Bu tanımlamalar, 12 tondan oluşan yaklaşık olarak 10.5 oktavlık bir 'nota'

sistemi, zamanlamayla ilgili mesajlar, program deęişim mesajları, *pitch wheel* ve benzeri önceden tanımlanmış bazı parametre mesajları ve genel kontrol mesajları içerir (5).

İletişim için MIDI standardını kullanan kontrol aygıtları, kontrol bilgilerini hedef aygıtta göndermek için, genellikle MIDI standardının genel kontrol mesajlarını kullanmaktadırlar. Bir kontrol mesajı, 3 byte'tan oluşur: Birinci byte, mesajın kontrol mesajı olduğunu ve hangi kanala gönderildiğini; ikinci byte, mesajın hangi kontrol parametresiyle (0'dan 127'ye kadar) ilgili olduğunu; üçüncü byte ise parametreye gönderilen değeri (0'dan 127'ye kadar) belirtir (5). Buna göre bir sentezleyicinin herhangi bir parametresi, 0 ile 127 arasında bir değer kabul edecek şekilde tasarlanmalıdır (iki kontrol mesajını gruplayarak parametre değer aralığını 0-16383'e genişletmek mümkündür). MIDI standardı dahilinde bazı kontrol mesajları için belirli fonksiyonlar tanımlanmıştır, fakat bu tanımlamalar her üründe tutarlı biçimde uygulanmamaktadır.

MIDI kontrolünün kontrol aygıtları tarafına yansıyan en yaygın uygulaması, aygıt üzerinde bulunan her düğme, tekerlek ya da kaydırma çubuğu için bir kontrol mesajı numarası tanımlanmasıdır. Aynı mesaj numarası, kontrol edilecek olan hedef aygıtta da istenen parametre için tanımlanır. Birden fazla boyutta çalışan arayüz bileşenleri, her boyut için ayrı bir kontrol mesajı numarası kullanabilir.

Tanımlanan bu sistemde, sentezleyici parametreleri ve onların kontrolünü sağlayacak bileşenler arasında önceden belirlenmiş özellikler bulunmadığı için, her iki tarafta da tasarımlar, MIDI standardının getirdiği genellemeyi benimsemeye mecbur kalmaktadır. Bu genellenenin, 1970-1990 arasındaki analog sentezleyici ve benzeri aygıtlar dönemine ait ürünlerin tasarımında yeterli bir kullanılabilirlik sağladığı söylenebilir. Nitekim, MIDI standardı da aynı dönemde, bu tür aygıtlara yönelik olarak hazırlanmıştır.

Fakat genel olarak kullanılabilirlik göz önüne alındığında, bu tür bir eşleşmenin, hem sentezleyicinin hem de kontrol mekanizmasının potansiyelini ortaya koymada eksikleri olması kaçınılmazdır. Özellikle en yaygın arayüz türlerinden biri olan piyano klavyesinin, sentezleme yöntemlerinin potansiyelinin ortaya çıkmasında sınırlayıcı bir rolü olduğu kolaylıkla gözlenebilir.

Geleneksel akustik enstrümanlara bakıldığında, ses üretim yöntemiyle kontrol mekanizması arasında güçlü bir bağ görülebilir. Bu bağın, söz konusu enstrümanın anlatım

dilini genişlettiđi, icrasında ustalığı desteklediđi ve icra deneyimini bütününde daha tutarlı kıldıđı söylenebilir.

1.4 Arayüzden Sentezleyiciye Doğru Tasarım

Buraya kadar sözü edilen hususlar ışığında, bu projede, bir fiziksel modelleme algoritmasını temel alan bir sentezleyici oluşturma fikrinden yola çıkılmış, önce bir kontrol mekanizması seçilmiş ve bu mekanizmayla birlikte çalışmaya yönelik bir sentezleme algoritmasının tasarlanması amaçlanmıştır.

Seçilen kontrol mekanizması, görsel tasarım ve ilgili alanlarda yaygın olarak kullanılan çizim tabletleridir. Piyasada kolayca bulunan bu tabletler, düz bir yüzey ve bu yüzey üzerinde çizim yapmak için kullanılan bir kalemden oluşur ve yatay/dikey hareket bilgilerine ek olarak kaleme uygulanan basıncı da algılayabilir.

Bu kontrol mekanizmasıyla eşleşmesi için sentezleyici tarafında da, sivri bir nesnenin, metal çubuk ve benzer nesnelere üzerinde sürtülmesiyle oluşan titreşimleri matematiksel olarak modelleyen, *banded waveguide* tekniđi üzerine kurulu bir algoritma kullanılmıştır.

Böyle bir eşleşmeyle, söz konusu fiziksel modelleme sentezleyicisinin basit hareketlere dayalı yalın bir kontrol mekanizması ile mümkün olduğunca etkin bir şekilde kullanılabilir hale getirilmesi amaçlanmıştır.

2. ARAYÜZ VE SENTEZLEME TEKNİĞİ SEÇİMİ

Canlı icranın ön planda olduğu bir arayüz ve sentezleme sisteminin kurulmasında, öncelikle bu sistemin kullanımı için esas olan bazı noktalar belirlenmiştir. Bu noktalar, önerilen sentezleme tekniğinin eş zamanlı çalışabilecek şekilde uygulanabilirliği; tekniğin uygulandığı yazılımın uzmanlık gerektirmeyecek şekilde kolay kullanılabilir olması; arayüz mekanizmasının hassasiyet sınırları, ergonomik özellikleri, ürün olarak erişilebilirliği; sentezleme tekniği ile arayüz mekanizmasının anlamlı biçimde eşleşmesidir.

2.1 İnsan-Bilgisayar Arayüzü Olarak Çizim Tabletleri

Çizim tabletleri, bilgisayarlara kağıt ve kalem kullanarak çizim yapma deneyimini katmak amacıyla ortaya çıkmış veri giriş aygıtlarıdır. Tipik bir çizim tableti, çizim yapılan plastik, düz bir yüzeyi ve bu yüzey ile kullanılmak üzere tasarlanmış bir kalemi (*stylus*) içerir.

1970'lerde endüstriyel ve bilimsel çizim işleri için üretilen, bugün kullanılan tabletlerin ataları sayılabilecek ilk tabletler, kalemin dikey ve yatay tellerden oluşan bir ızgara üzerindeki voltaj seviyelerinde yarattığı farkı ölçerek çalışmaktaydılar ve maliyetleri oldukça yüksekti. Bu tabletler yatay ve dikey konum bilgileri dışında veri girişi sağlamıyordu ve söz konusu teknolojinin olgunlaşmamış olması nedeniyle kullanımlarını zorlaştıran bazı özellikleri bulunuyordu.

Günümüzde profesyonel çizim işlerine yönelik üretilen tabletlerde, kalem tarafından yayılan ve sayısal veri içeren radyo dalgaları, plastik çizim yüzeyinin altında bulunan yüksek çözünürlüklü bir alıcı ızgarası tarafından algılanır ve edinilen veriler paketler halinde bilgisayara gönderilir. Kalemin ucunun bağlı olduğu yaya uygulanan basınç, kapasitans değişikliğine sebep olarak bu verinin kalem içindeki devre tarafından ölçülmesini mümkün kılar. Bazı kalemler üzerinde fare düğmelerine benzer işlevi olan düğmeler de bulunabilir. Kalem içinde bulunan bir pil, radyo dalgalarının yayılması için gereken enerjiyi sağlar.

Wacom firmasının ürettiği tabletlerde, çizim yüzeyinin altındaki ızgara dönüşümlü olarak radyo dalgası yayma ve algılama durumlarına geçer. Bu dönüşüm, yaklaşık olarak saniyede 50000 defa gerçekleşir. Kalemin içindeki elektromanyetik rezonans özellikli devre,

dalga yayma durumu sırasında ortaya çıkan enerjiyi kullanarak, kalemin ucuna uygulanan basınç, kalem üzerindeki düğmelerin durumu ve kalemin kimlik numarası gibi bilgileri, radyo dalgaları üzerine kodlayarak çizim yüzeyinin altındaki ızgaraya geri gönderir. Izgarayı oluşturan tellere ulaşan enerji seviyelerine dayanarak kalemin dikey ve yatay konumu hesaplanır (6).

Giriş seviyesinde üretilen tablet modellerinde, ortalama olarak yatay düzlemde 5000 dpi, dikey düzlemde 3000 dpi ve basınç için 512 seviyeli çözünürlük bulunur. Profesyonel kullanıma yönelik modellerde çözünürlük değerleri yükselmekte ve basınca ek olarak kalem ile çizim yüzeyi arasındaki açığı, kalemin kendi etrafında dönmesi gibi veriler de bilgisayara aktarılmaktadır.

Çizim tabletlerinde son yıllarda görülen gelişmelerden biri, ekranlarla birleşik hale gelmeleridir. ‘Tablet PC’ ve profesyonel çizim ekranlarında uygulanan bu birleşimde, çizim yüzeyinin altındaki ızgara, LCD düzeneğinin altına yerleştirilmekte, ekranın cam veya plastik ön yüzeyi, çizim yüzeyi olarak kullanılmaktadır. Böylece, kullanıcı ekranda gördüğü arayüz bileşenlerini doğrudan kontrol edebilmekte, göz ve el arasındaki koordinasyon önemli ölçüde artmaktadır.

2.1.1 Ergonomik Özellikler

Çizim tabletleri, kağıt ve kalem ile çizim deneyimini aktarmaya yönelik tasarımları nedeniyle, kullanılması kolay kontrol mekanizmalarıdır. Çizim tabletleriyle ilk kez karşılaşan, kalem tutma ve çizim yapma temel yetilerine sahip bir bilgisayar kullanıcısının bu mekanizmaya alışıp verimli bir şekilde kullanması, çoğu tabletle birlikte sunulan alıştırmaya yazılımlarının da yardımıyla, birkaç saat içinde gerçekleşebilmektedir. Olumlu ergonomik özellikleri nedeniyle, çizim tabletleri, *Repetitive Strain Injury* sorunu yaşayan kullanıcılara, bazı kaynaklar tarafından alternatif arayüz olarak tavsiye edilmektedir (7).

2.1.2 Ses Uygulamalarında Kullanım

Son 10 yıl içinde bilgisayarların ses üretiminde yaygın biçimde kullanılmasıyla birlikte, bilgisayarlarla kullanılacak amaca uygun arayüz arayışları da önem kazanmıştır. Çizim tabletlerinin satış fiyatlarının düşmesi, programlama açısından yazılımlara görece kolay

dahil edilebilmeleri ve çoğu MIDI temelli kontrol aygıtından daha iyi çözünürlükte veri sağlayabilmeleri, sınırlı da olsa ses uygulamalarında da kullanıma girmelerini sağlamıştır.

Çizim tabletlerinin ses uygulamalarındaki en yaygın kullanımı, tablet verilerinin MIDI kontrol mesajlarına dönüştürülerek sentezleme yazılımlarına gönderilmesini sağlayan aracı yazılımlarla birlikte görülmektedir (8), (9). Bu yazılımlarda, tabletin dikey ve yatay konumu, basınç seviyesi, kalem eğimi gibi bilgilerin her biri birer kontrol mesajına atanır, söz konusu veri değerleri 0-127 arasında değerlere ölçeklenir ve alıcı tarafa gönderilir. Fakat tabletlerin ses uygulamalarındaki potansiyelinin ortaya çıkarılmasında, MIDI çerçevesinin dışına çıkılması, sağladıkları hassaslık ve çözünürlüğün kullanımının mümkün hale gelmesi açısından önemlidir.

2.1.3 Ses Uygulamalarında Diğer Kontrol Aygıtlarıyla Karşılaştırma

MIDI temelli kontrol aygıtlarının özelliklerine ve eksikliklerine önceki bölümlerde değinilmişti. Bu noktalar göz önünde bulundurularak, var olan enstrümanları modelleme, yeni enstrüman veya sentezleme fikirleri önerme, yeni icra pratikleri ve mekanizmaları araştırma gibi alanları konu alan daha önce yapılmış birçok projede, projenin amacına yönelik bir kontrol mekanizmasının tasarlanması ve bu mekanizmanın hayata geçirilmesi, merkezî önem taşımaktadır.

Bu projelere örnek olarak, Tibet çanaklarını modelleyen ve modelin ‘doğal’ biçimde icrasını mümkün kılacak bir aygıt öneren HyperPuja projesi (10), elektronik sitar tasarımı projesi olan ESitar (11) ve Japon vürmalı çalgı geleneğine yeni bir arayüz öneren AoBachi (12) verilebilir.

HyperPuja’da tasarlanan elektronik ‘tokmak’ (*puja*) içindeki mikroçip, 10 bit’lik (1024 değer seviyesi) bir A/D çeviricisi kullanarak tokmağın çanak çevresindeki hızını, ivmesini ve çanak üzerine uyguladığı basıncı ölçüp, proje için tasarlanan basit bir protokol kullanarak kablosuz bağlantıyla proje yazılımını çalıştıran bilgisayara iletmektedir.

ESitar projesinde, basılan perde, teli çekme zaman bilgisi, parmak basıncı ve icracının kafasının eğimi verileri, sitar üzerinde çeşitli algılayıcılar kullanılarak MIDI mesajlarına dönüştürülerek bir bilgisayara gönderilmektedir.

AoBachi projesindeki çubukların içinde bulunan elektronik düzenek, iki boyutlu ivme ve açısal hız bilgilerini, Bluetooth üzerinden proje için belirlenen bir veri protokolü kullanarak proje yazılımının çalıştığı bilgisayara yollamaktadır. Bluetooth'un kullanılma sebebi olarak, veri aktarma hızı ve iletişim mesafesi gösterilmiştir.

Söz konusu projelerin ortak yanı olarak, ses üretimi için bir teknikten yola çıkılarak, bu tekniğe uygun kontrol aygıtı tasarlanması sayılabilir. Bu yaklaşımın olumlu sonucu, seçilen teknik için ileri derecede eşleşmeye uygun bir kontrol mekanizmasının kullanılmasının mümkün olmasıdır. Diğer yandan araştırma projelerine özel üretilen kontrol aygıtlarının seri üretim yapılmaması nedeniyle proje dışında çok sınırlı kullanım bulması, olumsuz sonuçlardan biri olarak gözlenebilir. Bunun dışında, donanım tasarımının maddi proje gereksinimlerini arttırması da yüksek bir olasılıktır.

Bu projede arayüz olarak, Wacom firması tarafından üretilen Volito2 çizim tableti kullanılmıştır. Bu tablet, 127.6 × 92.8 mm'lik bir alanda, milimetre başına 40 satır çözünürlükle kalem ucunun konumunu saptayabilmekte ve 512 seviyede basınç değerini algılayabilmektedir. 148.0 x 12.5 mm boyutlarında ve 12 gr. ağırlığındaki kalemin ucu, yaklaşık 2 mm hareket derinliğine izin vermektedir. Kalemin üzerinde fonksiyonları (sağ fare tuşu, orta fare tuşu, çift tıklama, v.b.) kullanıcı tarafından ayarlanabilen iki düğme bulunmaktadır. Tablet bilgisayara USB üzerinden bağlanmakta ve saniyede 100 defaya kadar veri yollamaktadır.

Seri üretilen ve kolayca bulunabilen bir ürün kategorisindeki çizim tabletlerinin bir ses uygulamasında arayüz olarak kullanılması önerisiyle, benzer projeler için sayılan olumsuz sonuçlardan kaçınılması; bu projede tasarlanan yazılımın, proje ve akademik ortamlar dışında da kullanımının mümkün olması amaçlanmaktadır.

2.2 İncelenen Akustik Enstrümanlar

Çizim tableti için bir sentezleme tekniğinin oluşturulmasında hem fiziksel özellikleri hem de kullanımları açısından bazı akustik enstrümanlar incelenmiştir.

2.2.1 Vibrafon

Vibrafon, A.B.D.'de ortaya çıkan ve modern perküsyon topluluklarının ve caz müziğinin önemli bir parçası haline gelen bir metal vurmali çalgıdır. Çalgının sesi, metal çubukların vuru içeren titreşimlerinin elektronik yöntemler veya bir çeşit rezonatör ile yükseltilmesi sonucu oluşur. Bir alüminyum alaşımından üretilen çubuklar, klavyeli bir çalgının tuşlarına benzer bir düzende, nodal noktalarından asılmış olarak dizilir. Söz konusu düzende piyanodaki siyah tuşlara denk gelen çubuklar, birden fazla bagetin kullanılmasını kolaylaştırmak için, diğer çubuklarla aynı seviyede bulunur. Bagetlerin ucunda genellikle lastik veya tel sargı kullanılır. Metal çubukların sönümlenme süreleri uzundur. Enstrümanda, rezonansı sağlayan düzeneğin kontrolü için, modern piyanonun sağ pedalına benzer işlevi olan bir kontrol mekanizması bulunur. Bu mekanizmanın pedalına basıldığında, çubukların sönümlenmesini arttıran keçe, çubuklardan ayrılır. Tipik bir vibrafonun ses aralığı, orta do'nun altındaki fa'dan başlayarak üç oktavdır (13).



Şekil 2.1 Vibrafon (14)

Vibrafonun önemli özelliklerinden biri, özgün *vibrato*'sudur. Bu özellik, rezonansın çubukların altına yerleştirilen bir ucu açık tüplerle sağlandığı modellerde, tüplerin ağızlarında bulunan hareketli metal disklerin açılıp kapanması sayesinde gerçekleşir. Metal diskler, bir motor tarafından döndürülen bir çubuğa bağlıdır. *Vibrato*'nun hızı, motorun hızının değiştirilmesiyle ayarlanabilir. Rezonans ve *vibrato*'nun *pickup* transdüktörler tarafından

sağlandığı ‘elektronik’ vibrafonlar da üretilmiştir. Bu vibrafonlarda *tremolo* ve genlik seviyesi kontrolleri bulunmaktadır (13).

2.2.2 Armonika

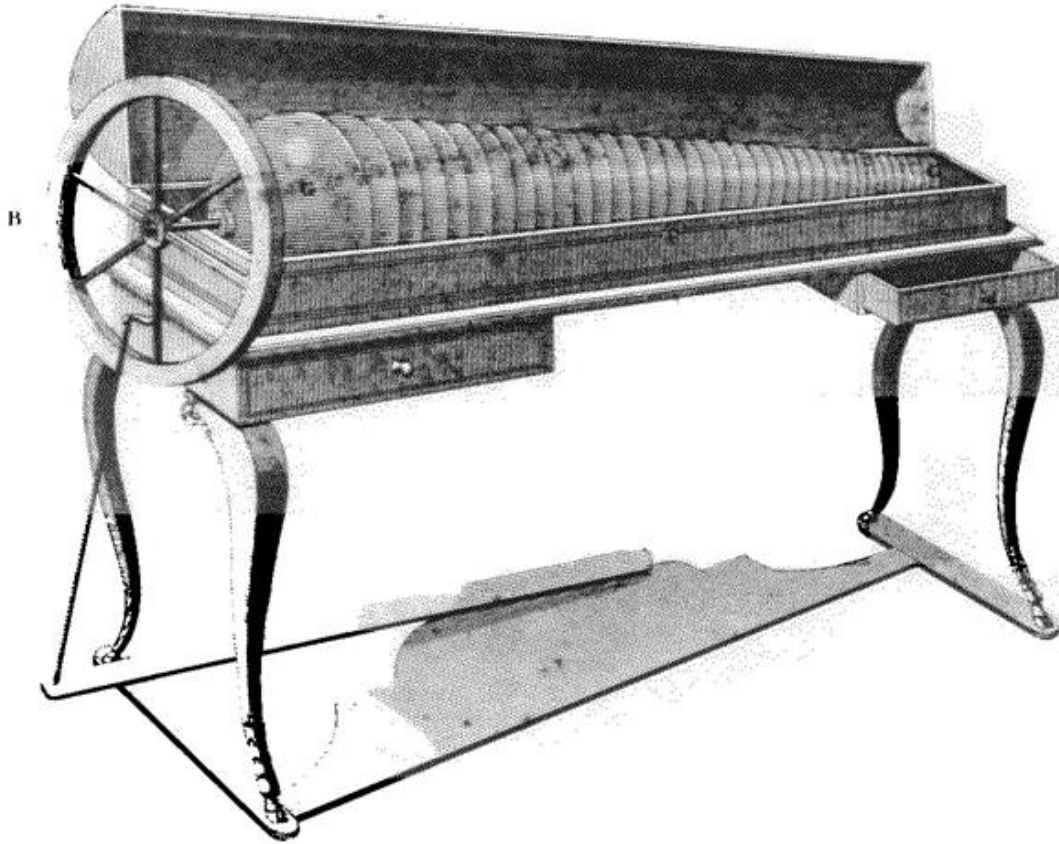
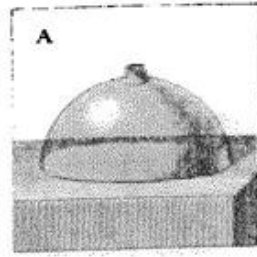
Armonika, cam ve benzeri sert maddelerden yapılan ‘kadeh’lerden oluşan, çan kategorisinde ele alınabilecek, kenarlarına sürtünme uygulandığında yaylı bir enstrümanın tellerine benzer tepkiyle ses üreten bir enstrümandır. Düşük kuvvetle vurmalı olarak kullanıldığında *tremolo* ve telin çekilmesine benzer etkiler de üretebilir (15).



Şekil 2.2 Armonika (16)

Camdan yapılmış nesnelerin Batı’da müzik içinde ciddi anlamda kullanıma girmesi 18. yüzyılda görülebilir. İlk örneklerde ses, cam kadehlerin yan yüzlerine vurularak üretiliyordu. Bu yöntem, bazı durumlarda yeteri kadar belirgin olmayan bir sonuç veriyordu.

İngiltere’de, kadehin kenarına parmak ucuyla sürtünme uygulayarak ses üretme yöntemi, ilk defa 1744’te Richard Pockrich tarafından kullanılmıştır. Pockrich’in kullandığı kadehler büyüklüklerine göre sınıflandırılmıştı ve tonlarının ayarlanması için içlerine su konmaktaydı. Önemli armonika icracılarından Ann Ford, 1761’de bu enstrüman için bilinen ilk alıştırma kitabını yayınladı. Bu alışırmalarda, nemli parmak uçlarının kullanılması isteniyor ve parmak ucuyla kadehin kenarına ve yan yüzüne uygulanan basınç seviyeleri detaylı olarak tanımlanıyordu (15).



Şekil 2.3 Benjamin Franklin'in Armonika tasarımı (17)

Daha sonra Benjamin Franklin, kadehlerin 'kâse' kısımlarını, yatay bir çubuk üzerinde dizerek yeni bir armonika tasarımı oluşturdu. Yatay çubuk bir pedala bağlı olarak döndürülebiliyordu. Kadehlerin büyüklüklerinin dikkatle seçilmesi, su ile yapılan akorddan daha iyi bir sonuç vermişti. Kadeh kenarlarının birbirine yakın olması sayesinde de, icracının akorları ve hızlı hareket eden melodileri çalması daha kolay hale gelmişti. Buna daha sonra, kadehlerin kenarlarını otomatik olarak ıslatabilen bir mekanizma eklendi. Sonraki dönemlerde titreşim yaratma mekanizması olarak parmak ucunu kullanmamak için çeşitli değişiklikler denendi, fakat hiçbir yöntem parmak ucuyla sağlanan tınıyı üretmekte başarılı olmadı (15).

2.2.3 Tibet Çanağı

Tibet çanakları, metal alaşımlarından veya camdan yapılan, yarım küreden silindire kadar değişebilen şekillerde üretilen enstrümanlardır. Çanak vurmaları çalgı olarak kullanıldığında çan sesine benzer bir tını üretmektedir. Tınının en önemli özelliği, frekansları birbirine yakın olan titreşim modlarının yarattığı vurulardır. Tını niteliği büyük ölçüde çanağın şekline, üretim maddelerine ve boyutlarına bağlıdır (18).



Şekil 2.4 Tibet çanağı ve tokmağı (*puja*) (19)

Tibet çanağının yaygın kullanım yöntemi, çanağın kenarına tahta bir tokmağın sürtülmesidir. *Puja* adı verilen bu tokmağın, icracı tarafından çanağın çevresi boyunca çeşitli hızlarda ve kuvvetlerde sürtülmesi, sönümlenmesi uzun zaman alan çınlamalı bir ses yaratır (18).

2.3 Sentezleme Tekniği Olarak Fiziksel Modelleme

Ses üretim tekniği olarak fiziksel modelleme, sentezlemeyi oluşturan aşamaların fiziksel olaylar üzerine kurulu olduğu algoritmaların kullanıldığı bir yöntemdir. Böyle bir algoritmanın oluşması için, öncelikle bir enstrümanın veya daha basit bir nesnenin titreşim mekanizmaları matematiksel olarak tanımlanır. Elde edilen modelde, hedeflenen uygulamanın sınırlarına göre çeşitli indirgemeler yapılabilir. Bunun sonucunda, değişkenleri ('parametre'leri) modellenen nesnenin ya da titreşim mekanizmasının fiziksel özelliklerine denk gelen bir algoritma elde edilir. Bu algoritma, modellenen enstrümanı gerçekçi biçimde canlandıracak parametrelerle çalıştırılabileceği gibi, enstrümanın fiziksel sınırlarının ötesine geçen parametrelerle çalıştırılarak 'imkânsız' durumları da canlandırabilir.

Fiziksel modellemenin temelindeki kavramlar 18. yüzyıla kadar geriye gitmektedir (20). Fakat tasarlanan algoritmalar, genellikle işlem hacmi yüksek uygulamalar ortaya çıkarmaktadır. Bu yüzden fiziksel modelleme tekniklerinin özellikle eş zamanlı kullanımı, ancak yakın geçmişte, mikroçiplerin ve diğer ilgili donanımların yeterli hızlara ve kapasitelere erişmeleriyle mümkün olmuştur.

Bunun yanı sıra, fiziksel modelleme uygulamalarının sunduğu kontrol edilebilirliğin arayüzler tarafından yeterince temsil edilmemesi de, bu uygulamaların başarısını olumsuz yönde etkilemiştir (21), (22). Fiziksel modelleme algoritmalarının en önemli özelliklerinden biri olarak, ses oluşma sürecini somut parametreler yoluyla, modellenen sistemin doğasına uygun şekilde yönlendirmeyi sağlamaları sayılabilir. Modelin kullanımında gerçekçilik amaçlanıyorsa, bu olanaktan yararlanmak bir zorunluluk halini alır. Buna bağlı olarak, fiziksel modelleme uygulamalarının icrasında, modelle iyi eşleşebilecek bir kontrol aygıtının kullanılması büyük önem taşır.

Çizim tabletleriyle çalışmak üzere bir modelin seçilmesinde, öncelikle tabletlerin mekanik çalışma şekillerine benzetilebilecek titreşim mekanizmaları gözden geçirilmiş, sonra

da bu mekanizmalara benzer özelliklere sahip bazı akustik enstrümanlar incelenmiştir. En sonunda, oluşturulan modeli iyi temsil edebilecek bir algoritma türü saptanmıştır.

2.3.1 Çizim Tabletlerinin Mekanik Özellikleri İçin Temel Model

Bir fiziksel modelleme çalışması olarak bu proje dahilinde her ne kadar nesnel ölçümlere ve kanıtlanmış teorilere dayanılsa da tasarlanan modelde ve bu modelin uygulanmasında, bazı genellemeler, indirgemeler ve amaçlanan kullanıma yönelik uyarlamalar yapılmıştır.

Tablet kontrol mekanizmasının titreşim mekanizmasına dönüştürülmesi en yalın şekilde ele alınmış, kalemin sivri uçlu bir nesne olarak, genel olarak düz, fakat küçük ölçekte düzensiz pürüzlerin bulunduğu yüzeyleri olan bir nesne üzerinde sürtünmeyle enine (*transverse*) dalgalardan oluşan titreşimler yarattığı bir temel model benimsenmiştir.

Tablet çizim yüzeyinin tek bir parçadan oluşmasına karşın, birden fazla temel frekansta titreşim oluşmasını olanaklı hale getirmek için, çizim yüzeyinin birden fazla parçaya bölünebilir olması, birden fazla nesneyi temsil etmesi düşünülmüştür. Bu durumda fiziksel modelin yapı taşı, bir çubuk olarak görülebilir.

Çubukların her birinin titreşim durumu diğerlerinden bağımsızdır. Çubukların uç durumları ve sönümlenme durumu gibi bazı özelliklere bağlı olarak değişen titreşim modları ve genelleştirilmiş rezonans dereceleri bulunmaktadır. Sivri uçlu nesne ile bir çubuğa uygulanan kuvvetin büyüklüğü ve uygulama bölgesi titreşimin özelliklerini etkilemektedir.

2.3.2 Temel Model İçin Algoritma Seçimi

Fiziksel modelleme algoritmalarının çoğu, dalga denkleminin modellenen enstrümana uygulanmış bir halinin çözümü üzerine kuruludur (20). Enstrümanın fiziksel yapısını temsil eden sanal bir 'ızgara'yı oluşturan bütün noktalar için çözüldüğü algoritmalar, işlem hacminin çok yüksek olduğu algoritmalarlardır.

Bir başka yaklaşımda, dalga denklemi, herhangi bir ses taşıyıcı ortam için enine (*transverse*) dalgaları belirleyecek şekilde genelleştirilerek çözülür ve bu dalgaların ortam boyunca yayılması modellenir. *Waveguide* (dalga yönlendiricisi) adı verilen bu algoritma

türünün işlem hacmi bazı fiziksel modelleme algoritmalarına göre oldukça düşüktür ve yayla çalınan çubuk, armonika ve çan benzeri enstrümanların modellenmesinde kullanılmaya uygundur (23).

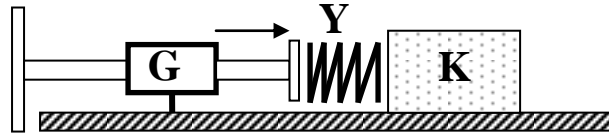
Dalga yönlendiricisi, bir dalganın yayıldığı herhangi bir ortam olabilir. Dalga kaynağı, bir telin çekilmesi, bir tel ile yay arasında sürtünme, bir ucu kapalı bir boruda hava basıncının değişmesi gibi çeşitli mekanizmaların modelleri tarafından sağlanabilir. Dalganın yayılımı, ortam yoğunluğu değişmediği sürece aynı kalır. Yayılma hatları, *delay* (gecikme) hatları ile temsil edilebilir. Ortam değişikliğinin yarattığı hareket dağılımları ve yansımalara bağlı oluşan dalga hızı ve frekanslarındaki farklar, çeşitli filtreler ile modellenebilir. Söz konusu *waveguide* bileşenlerinin çoğunun sonuç üzerinde değişikliğe sebep olmadan sıralarının değiştirilebilir olması, bu bileşenlerin çeşitli noktalarda toplanarak sayılarının azaltılmasına ve işlem gereksiniminin küçültülmesine imkân verir. Güncel teknolojik olanaklarla, *waveguide* algoritmaları canlı icralar için eş zamanlı olarak kolaylıkla uygulanabilmektedir.

3. TEORİ VE ALGORİTMA

Bu projede tasarlanan sentezleyicide, sürtünme prensiplerinden yola çıkan, modellenen nesnenin titreşim modlarına uygun frekanslarda dalgalar oluşturan ve çalışması boyunca sürekli olarak kullanıcı tarafından yönlendirilebilen bir süreç kullanılmıştır. Bu süreç, temelinde *bandpass* (kuşak geçirimli) filtreler ve *delay* hatları bulunan bir *banded waveguide* algoritması ile modellenmiştir.

3.1 Sürtünme

Çizim tableti için seçilen temel modeldeki düz yüzey ile sivri uçlu nesne arasındaki sürtünme, *stick-slip* (yapışma-kayma) olarak tanımlanabilir (24). Keman yayı ve teli, bisiklet freni ve tekerleği ile tektonik plakalar arasında da aynı olay görülür.



Şekil 3.1 Sembolik *stick-slip* sürtünme mekanizması

Stick-slip sürtünme, iki yüzeyin dönüşümlü olarak birbirlerine yapışma ve kayma durumlarına geçmeleridir. (Bu sürecin yapısı Şekil 3.1’de görülebilir. **G** nesnesi bir itici kuvvet mekanizmasını, **Y** nesnesi bir yayı, **K** nesnesi de yatay olarak itilen bir kütle temsil etmektedir.) Yapışma aşamasında, statik sürtünme katsayısı, kinetik sürtünme katsayısından daha büyüktür (**Y** üzerinde biriken kuvvet **K**’yi hareket ettirmek için yeterli değildir). Yüzeyle statik sürtünmeye karşı gelebilecek bir kuvvet uygulandığında sürtünmenin kayma durumuna geçmesi (sembolik sistemin üzerinde bulunduğu yüzeyin, **Y** tarafından **K**’ye uygulanan güce karşı koyamama noktasına gelmesi), sürtünme hızında ani yükselmeye sebep olur.

Kısa süreli de olsa, uzun süreli de olsa, kayma durumu düzensizlik yaratan bir durumdur (24). Anlık kaymalar, kapı menteşelerindeki gıcırtilarda görüldüğü gibi düzensiz aralıklarla ya da kemanın *spiccato* çalındığı durumlardaki gibi tekrarlı olarak ortaya çıkabilirler. Uzun süreli, aralıksız kayma ise, çok çeşitli türlerde ve frekans spektrumlarında titreşimlere sebep olabilir.

Herhangi bir sürtünme sisteminde oluşan titreşimler, sürtünme arayüzündeki kuvvetlerin dalgalanmasına bağlı olarak gelişir. Bu kuvvetler de, söz konusu arayüzün özelliklerine ve dışarıdan uygulanan kuvvetlere bağlıdır. Sistemin başka bileşenlere bağlı olduğu durumlarda, titreşimlerin oluşmasında bu bileşenlerin de etkisi olacağı için, sürtünme sistemi bu bileşenlerle birlikte ele alınmalıdır.

Birbirine sürtülen iki nesnenin yaratacağı titreşimlerin türü, sürtünme arayüzündeki kuvvetlerin büyüklüklerine bağlıdır. Düşük kuvvetler uygulandığında, titreşimler yerel sürtünme bölgesinde ve her sürtünme bileşeninin öz titreşim frekansında oluşur. Pürüzlü yüzeyler arasında oluşan sürtünmelerdeki düşük kuvvetli kayma hareketleri, pürüzlerin birbirleriyle temas haline gelmesi sonucu küçük düzensizlikler yaratır. Bu düzensizlikler de bileşenlerin öz titreşim frekanslarında titreşimler oluşturur.

Pürüzlü yüzeye sahip bir çubuğa, uzunluğu boyunca başka bir pürüzlü nesneyle sürtünme uygulandığında, pürüzler üzerinde ve etrafında oluşan kuvvetler, hem normal hem de teğet yönlü bileşenler içerdiği için, boyuna (*longitudinal*) ve enine (*transverse*) dalgalar oluşur.

(25)'te metal vurmali çalgılardan alınan çubuklara kontrabas yayı sürtülmesi sonucu oluşan titreşimlerle ilgili bazı bulgular açıklanmıştır. Bu bulgulara göre, titreşimin tepe noktasındaki enerji seviyesi, yay sürtünme hızıyla lineer orantılı olarak artmaktadır. Bu orantılı artış göz önüne alındığında, enerji yayılımı sürtünme kuvvetinden bağımsız görünmektedir. Titreşimlerin spektral merkezinin, sürtünme hızı ve kuvvetiyle bir bağlantısı olduğu gözlenmemiştir. Yaylı çalgıların tellerinde görülen aksine, kuvvetin artması sonucu merkezin yukarı kayması görülmemiştir. Temel frekans ölçümleri, hız ve kuvvet değişimlerinin sonucunda belirgin bir eğilim göstermeyen küçük çaplı dalgalanmalar yaratmıştır. Titreşim başlama süresi de, sürtünme kuvvetinin artmasıyla orantılı olarak azalmaktadır; sürtünme hızının ise herhangi bir etkisi gözlenmemiştir.

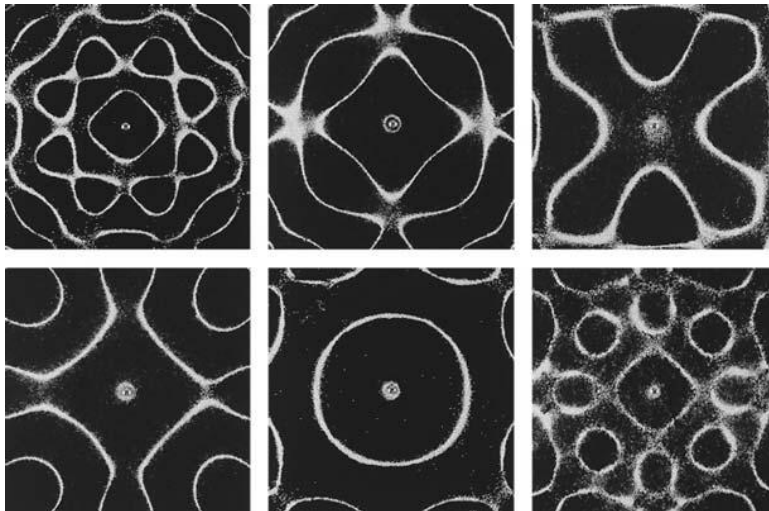
Sürtünme süreci için çok sayıda matematiksel tanımlama ve model bulunmaktadır (26). Bu projede, düzensiz pürüzlü yüzeye sahip çubuklar ile sivri uçlu bir nesne arasındaki sürtünme, beyaz gürültü (*white noise*) çıkış noktalı olarak ve sözü edilen metal çubuk ve kontrabas yayından elde edilen bulguları esas alarak modellenmiştir.

3.2 Öz Titreşim Modları

Bir nesnenin mekanik titreşim halinde bulunması için iki gereksinim sayılabilir: Nesnenin gerginlik veya yay benzeri bir niteliği olması ve titreşim hareketinin denge durumunun ötesine geçmesini sağlayacak mukavemet özelliğinin bulunması. Enerji açısından bakıldığında, bir nesnenin potansiyel ve kinetik enerji saklayabilme ve enerji kaybedebilme mekanizmalarına sahip olması gereklidir (27).

Öz titreşim modları, bir titreşim sistemini oluşturan bütün bileşenlerin aynı sinüzoidal hareket içinde bulunduğu durumlardır. İnce çubuk gibi bir nesne için bu modlar, nesnenin yapım malzemeleri, boyutları ve şekli gibi özelliklerine bağlı olarak ortaya çıkar.

Titreşim modlarıyla ilgili bazı önemli deneyler, Çek fizikçi Ernst Chladni (1756-1827) tarafından yapılmıştır. Bu deneylerde, ortasından sabitlenen ve üzerinde çeşitli konumlarda nodal noktalar yaratılan bir metal plaka, keman yayıyla titreştiriliyor ve üzerine ince taneli kum serpiliyordu. Titreşim hareketi nedeniyle, kum taneleri hareket olmayan nodal noktalarda toplanıyor, titreşim modlarının plaka üzerindeki bölgeleri görünür hale geliyordu. Bu bölgelerin oluşturduğu motifler, Chladni motifleri olarak anılmaktadır (28).



Şekil 3.2 Metal plaka üzerinde kum tanelerinin oluşturduğu Chladni motifleri (29)

Titreşim modlarını gözlemlemenin başka bir yolu da holografik interferometridir (30). Bu işlemde, bir lazer ışını iki ana ışına bölünür ve ışıklardan biri ölçüm yapılan nesne üzerinden yansıtılırken, diğeri de doğrudan referans ışını olarak kullanılır. Her iki ışın da bir film üzerine düşürülür. Film, fotoğraf banyosuna benzer bir işlemden geçtikten sonra, nesnenin nodal ve antinodal alanlarını gösteren bir hologram oluşur.



Şekil 3.3 Titreşim halindeki bir çanın hologramı (30)

Gergin tellerde görülen enine titreşimlerin benzeri, çubuklarda da görülebilir. Tellerde görülenin aksine, gerginlik olmaksızın, çubuğun normal haline dönmesini sağlayan kuvvetler, çubuğun kendi elastik kuvvetleri tarafından sağlanır (27). Uzunluğuna nazaran ince bir çubuğun enine titreşim modları, 1 boyutlu Euler-Bernoulli modeliyle hesaplanabilir (31):

$$\frac{\partial^2}{\partial x^2} \left(EI \frac{\partial^2 y}{\partial x^2} \right) + \left(\rho A \frac{\partial^2 y}{\partial t^2} \right) = f(x, t) \quad (3.1)$$

E , maddenin elastik Young modülünü; I , dik kesitteki mukavemet momentini (*cross-sectional moment of inertia*); ρ , madde yoğunluğunu; A , dik kesit alanını belirtmektedir. Tek bir frekans için dalga yayılım hızı (v), $f(x, t) = 0$ ve $a = \sqrt{EI/\rho A}$ olduğu takdirde

$$v = \sqrt{a\omega} \quad (3.2)$$

formülüyle hesaplanmaktadır (31). Buna göre, bir nesne içinde enine dalgaların hızları, frekanslarına bağlı olarak değişmektedir. Nesne içinde oluşan genel titreşimin şekli de bu hız farkları sayesinde ortaya çıkmaktadır. (3.1) denklemi, sabit katsayıların olduğu ve dış kuvvetlerin olmadığı durum için

$$y = e^{j\omega t} (Ae^{kx} + Be^{-kx} + Ce^{jkx} + De^{-jkx}) \quad (3.3)$$

şeklinde çözülebilir (31). Bu denklemde, dört sınır koşulu (*boundary condition*) bulunmaktadır. Sınır koşulları, çubuğun uç durumlarına göre belirlenmektedir; uçlardan her biri, serbest, sabitlenmiş veya menteşe destekli olabilir.

İki ucu serbest bir çubuğun öz titreşim modlarının frekansları

$$f_n = \frac{\pi K}{8L^2} \sqrt{\frac{E}{\rho}} [3.011^2, 5^2, 7^2, \dots, (2n + 1)^2] \quad (3.4)$$

formülüyle hesaplanabilir (27). n hesaplanan titreşim modunun sıra sayısını, K dönme yarıçapını (*radius of gyration*; dikdörtgen prizma şeklindeki çubuklar için $K = h/\sqrt{12}$), L çubuk uzunluğunu belirtmektedir. Denklem $[3.011^2, 5^2, 7^2, \dots, (2n + 1)^2]$ bölümü, mod frekanslarının temel frekansa olan oranlarını belirler. Bu oranların ilk 10 tanesi Tablo 3.1’de listelenmiştir. Oranların armonik (temel frekansın tam sayı katları) olmadığı görülebilir.

Bir ucu serbest, bir ucu sabitlenmiş bir çubuğun öz titreşim modlarının frekansları

$$f_n = \frac{\pi K}{8L^2} \sqrt{\frac{E}{\rho}} [1.194^2, 2.988^2, 5^2, \dots, (2n - 1)^2] \quad (3.5)$$

formülüyle hesaplanır (27). Denklem $[1.194^2, 2.988^2, 5^2, \dots, (2n - 1)^2]$ bölümü, frekans oranlarını belirler. Oranların ilk 10 tanesi Tablo 3.2’de görülebilir.

İki ucunda menteşeli destekler bulunan bir çubuğun öz titreşim modlarının frekansları

$$f_n = \frac{\pi K}{8L^2} \sqrt{\frac{E}{\rho}} m^2; \quad m = 1, 2, 3, \dots \quad (3.6)$$

formülüyle hesaplanır (27). Bu denklemde de frekans oranları m^2 bölümünde belirlenmektedir. Oranların ilk 10 tanesi Tablo 3.3’te görülebilir. Bu oranlar, temel frekansın tam sayı katlarıdır.

Titreşim Modu	Temel Frekansa Oranı (:1)	Titreşim Modu	Temel Frekansa Oranı (:1)
1	1	6	18.638
2	2.757	7	24.814
3	5.404	8	31.873
4	8.933	9	39.813
5	13.345	10	48.636

Tablo 3.1 İki ucu serbest çubuk için titreşim modlarının frekans oranları

Titreşim Modu	Temel Frekansa Oranı (:1)	Titreşim Modu	Temel Frekansa Oranı (:1)
1	1	6	84.874
2	6.263	7	118.544
3	17.536	8	157.824
4	34.371	9	202.717
5	56.817	10	253.220

Tablo 3.2 Bir ucu sabitlenmiş çubuk için titreşim modlarının frekans oranları

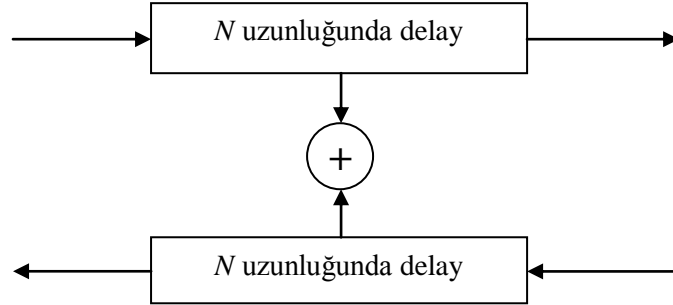
Titreşim Modu	Temel Frekansa Oranı (:1)	Titreşim Modu	Temel Frekansa Oranı (:1)
1	1	6	36
2	4	7	49
3	9	8	64
4	16	9	81
5	25	10	100

Tablo 3.3 İki ucu menteşe destekli çubuk için titreşim modlarının frekans oranları

3.3 “Banded Waveguide” Temelli Algoritma

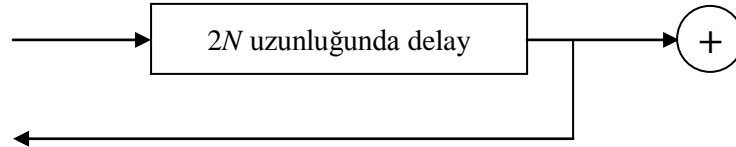
Waveguide algoritmaları, kayıp oranı düşük olan dalga yayılım ortamlarındaki hareket eden dalgaları (*traveling wave*), ortam geometrisini hesaba katarak düşük işlem gereksinimiyle modellemek için kullanılan, temelinde *delay* hatları, çeşitli filtreler ve modellenen nesneye veya enstrümana göre gereken diğer bileşenlerin bulunduğu sentezleme algoritmalarıdır.

En basit *waveguide* algoritması çift yönlü bir *delay* hattından oluşur (20). Fiziksel nesnelere hareket eden dalgaların doğrudan ölçümü mümkün olmadığı için, basınç ve hız gibi fiziksel değişkenlerin hesaplanması, dalga bileşenlerinin toplamının ölçülmesiyle yapılır.



Şekil 3.4 Temel *waveguide* akış şeması

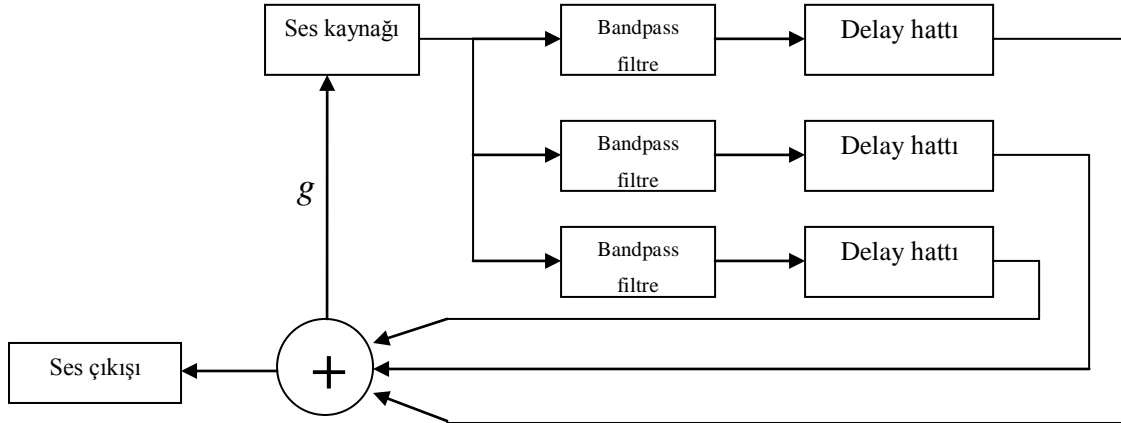
Waveguide algoritmalarını verimli kılan özelliklerinden bazıları, çoğu durumda algoritmayı oluşturan bileşenlerin sırasını değiştirmenin sonuç üzerinde değişiklik yaratmaması (*commutativity*) ve bileşenlerin türlerine göre gruplanarak genel etkilerini yaratacakları tekil noktalarda toplanabilir olmalarıdır (20).



Şekil 3.5 Bileşenlerin toplandığı (*commuted*) temel *waveguide* şeması

Temel *waveguide* algoritması, çubuk gibi 1 boyutlu titreşim nesnesi olarak sınıflandırılabilir nesnelere basit bir modelinin oluşturulması için yeterlidir (32). Çubuklar için enine dalgaların modellenmesinde, her frekanstaki dalga için hızının ayrı hesaplanması gerektiği önceki bölümde belirtilmişti. Bunun sonucunda tek bir çubuk modeli için çok büyük sayıda *delay* hattı ve diğer bileşenlerden oluşan *waveguide* zinciri gerekmektedir, işlem gereksinimi büyümektedir. Söz konusu modelin tipik kullanımında önemli olan spektral verinin, öz titreşim modlarının frekanslarında olması, *waveguide* zincirlerinin sadece bu frekanslara yönelik kurulduğu modellerin tasarlanmasını mümkün kılar.

Titreşim modlarının frekanslarına yönelik zincirlerin kurulması, zincire *bandpass* filtrelerin eklenmesiyle yapılmaktadır. Filtrelerin merkez frekansları, mod frekanslarına ayarlanır. Bu da her bir filtreden geçen merkez dışı frekansın yayılım hızının, merkez frekansın yayılım hızına eşit olduğu indirgemesinin yapıldığı anlamına gelir. Bu indirim sayesinde, algoritmanın işlem gereksinimi canlı kullanıma uygun hale getirilmektedir (23).

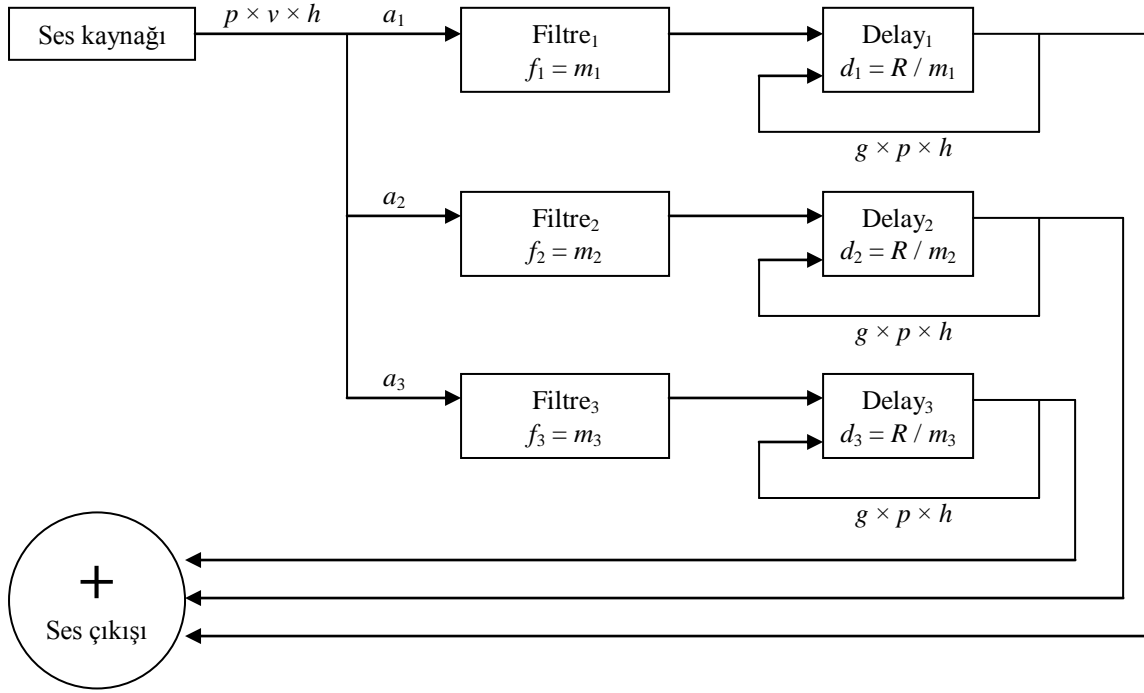


Şekil 3.6 Temel *banded waveguide* akış şeması

Şekil 3.6'daki şemada ses kaynağından elde edilen dalganın önce filtreler sayesinde frekans bantlarına ayrıldığı, her zincirin frekansına uygun uzunlukta ayarlanmış *delay* hatlarından geçtiği ve bu sinyallerin sonra tekrar toplandığı görülebilir. Toplama aşamasının sonunda sinyalin bir kopyası çıkışa gönderilirken, bir kopyası da g katsayısıyla çarpılıp *waveguide* zincirlerine tekrar beslenmek üzere ses kaynağına yollar. g katsayısı, enerji kaybını simgeler ve 1 ile 0 arasında bir değere sahiptir. g büyüdükçe, dolaşım halindeki dalgaların sönümlenmesi gecikir.

Bu projede tasarlanan modelin akış şeması, algoritmayı yönlendiren parametrelerle birlikte Şekil 3.7'de görülebilir. Şemada p , sürtünme kuvvetini (kalem basıncını); v , sürtünme hızını (kalem hareketinin hızını); h , sürtünmenin dikey konumuna bağlı olarak uç durumlarının yarattığı sönümlenmeyi; a_n , titreşim modu n için kullanılan genlik oranını; f_n , titreşim modu n için kullanılan filtrenin merkez frekansını; m_n , titreşim modu n 'nin frekansını; d_n , titreşim modu n için kullanılan *delay* hattının uzunluğunu; R , sistemin örnekleme hızını (*sample rate*); g , *delay* hatları için kullanılan geri besleme (*feedback*) katsayısını temsil

etmektedir. (Parametre katsayılarının hesaplanmaları ve birbirleriyle olan ilişkileri basitleştirilerek gösterilmiştir. Detaylar için bkz. bölüm 4.2.3)



Şekil 3.7 Tasarlanan *banded waveguide* modelinin akış şeması

Şemadaki ses kaynağı, bir önceki bölümde açıklandığı üzere beyaz gürültü üreten bir fonksiyondur. Bu fonksiyondan alınan sinyalin genliği, sürtünme kuvveti, sürtünme hızı ve uç durumlarının yarattığı sönümlenme oranlarına göre ölçeklenir. Sinyalin genliğinin 0'ı aşabilmesi için, hem sürtünme kuvvetinin hem de sürtünme hızının 0'dan büyük olması gerekmektedir. Uç durumlarının yarattığı sönümlenmenin etkisi sınırlandırılmıştır; sinyalin genliğini 0'a yaklaştırabilir fakat eşitleyemez.

Bundan sonraki aşamalarda, sinyal her titreşim modu için ayrı bir yol izleyerek işlem görür. Öncelikle genel spektral karakteri yaratmak için, titreşim modunun genlik katsayısıyla çarpılır. Bunu *bandpass* filtre izler.

Kullanılan *bandpass* filtre, 4. dereceden (-24 dB/oktav) birer *lowpass* ve *highpass* filtrenin arka arkaya kullanılmasıyla elde edilen bir filtredir. *Lowpass* ve *highpass* filtrelerin merkez frekansları, *bandpass* filtrenin merkez frekansı olan titreşim modu frekansına eşittir. Bu sayede istenen mod frekansına ek olarak düşük miktarda merkez dışı frekanslar da bir sonraki aşamaya gönderilir.

Filtreden sonra, bir *delay* hattı bulunmaktadır. Bu hattın uzunluđu, sistemin çalıştığı örnekleme hızının, titreşim modu frekansına bölümüne eşittir. Bu da söz konusu enine dalganın, titreşim nesnesi içinde durağan dalga (*stationary wave*) oluşturmasına denk gelmektedir. *Delay* hattının çıkışının bir kopyası ses çıkışına gönderilirken, bir kopyası da hattın girişine geri gönderilir. Geri dönen sinyalin genliđi, genel geri besleme katsayısına, sürtünme kuvvetine ve uç durumlarının yarattığı sönümlenme oranlarına bađlı olarak azaltılır. Sürtünme kuvvetinin bu noktada kullanılması, bir yandan ses kaynağından elde edilen sinyalin genliđini artırırken, bir yandan da sistem içinde dönmekte olan sinyalin sönümlenmesini hızlandırdığı anlamına gelir. Sürtünme hızının 0 olduđu durumlarda sürtünme kuvvetinin 0'dan büyük olması, sistemin genelinde yeni sinyal oluşmazken sönümlenmenin hızlanmasına sebep olur. Geri beslemenin etkisi, sinyaldeki merkez dışı frekansların giderek azalması ve merkez frekansın daha belirgin hale gelmesi olarak görülür.

Son olarak, *delay* hatlarından çıkan sinyallerin hepsi toplanarak ses çıkışına gönderilir. Bu sürecin geneline bakıldığında, frekans spektrumu zengin, düzensiz bir durumdaki sinyalin, frekans spektrumu sadeleşerek öz titreşim modlarının frekanslarına toplanmış, daha düzenli bir durumdaki sinyale dönüştüğü görülebilir.

4. UYGULAMA

Önceki bölümlerde açıklanan kavramların uygulaması olarak, Plaka adlı sentezleme yazılımı geliştirilmiştir. Bu süreçte, söz konusu kavramların doğrultusunda, çizim tabletleriyle birlikte çalışmayı merkezine koyan, canlı icraya uygun, temelindeki algoritmayı mümkün olan en kullanışlı şekilde yönlendirmeyi sağlamaya yönelik tasarım ilkeleri benimsenmiştir.

Plaka yazılımının anlatıldığı bu bölümde, kaynak kodundan doğrudan yapılan alıntılar ve benzer programlama ifadeleri, farklı yazı tipiyle gösterilmiştir (ör. `TPaintBox`). Programlama kodlarına yapılan göndermeler, köşeli ayraç içinde kodun bulunduğu dosya ismiyle birlikte kıvrımlı ayraçlar içinde numaralı olarak gösterilmiştir (ör. `[tabletoscForm.pas {1}]`). Dosya isimlerinin önceden belirtildiği durumlarda sadece referans numarası kullanılmıştır. Bu numaralar yazılım kaynak kodu içinde aratılarak gönderme yapılan kod bölümüne ulaşılabilir. (Yazılım kaynak kodu için bkz. Ek-1)

4.1 Altyapı

Yazılım geliştirme ortamı olarak, Borland firmasının Delphi 7 yazılımı kullanılmıştır. Delphi, Windows tabanlı, Pascal programlama dilinden türetilmiş kendi programlama dilini kullanan, nesne yönelimli (*object-oriented*), genel amaçlı bir yazılım geliştirme ortamıdır.

Delphi'de bulunan VCL (*Visual Component Library*; Görsel Bileşen Kütüphanesi), işletim sistemi API'lerinin (*Application Programming Interface*; Uygulama Programlama Arabirimi) fonksiyonlarını anlaşılması daha kolay nesnelere olarak programcıya sunarak, yazılım geliştirme sürecini hızlandırır. Bu nesnelere birçoğunun görsel temsilinin olması, nesne fonksiyonlarının öğrenilme süresini azaltır. VCL'de bulunmayan fonksiyonlar için dışarıdan bileşen (*component*) paketleri eklemek de mümkündür.

Delphi derleyicisi (*compiler*), Delphi projelerinin kaynak kodunu optimize ederek x86 koduna dönüştürür. Tipik durumlarda bir uygulamanın tamamı tek bir *Executable* dosyada toplanır (uygulamanın dağıtımı için sadece bu dosyanın verilmesi yeterlidir). Optimizasyonlar, yeni işlemcilerin yeteneklerinden tamamen faydalanmamakta, örneğin SSE (*Streaming SIMD Extensions*) gibi işlemci fonksiyonlarını içermemektedir. Fakat bu

eksikliklerin, proje kaynak kodu içerisinde doğrudan *assembly* kodu kullanılmasıyla giderilmesi mümkün olmaktadır.

4.1.1 Ses Donanımına Erişim

Ses donanımına erişim için Steinberg firmasının tasarladığı ASIO (*Audio Stream Input Output*) arabirimi kullanılmıştır. ASIO arabirimini kullanan uygulamalar, uygun ses donanımlarıyla birlikte kullanıldıklarında, işletim sisteminin ara katmanlarını atlayarak ses verisinin üretilmesi ile ses donanımından çıkması (veya verinin ses donanımından sisteme girmesi ile uygulamaya ulaşması) arasında geçen zamanı önemli ölçüde azaltabilir, stereo dışındaki yapılarda ses kanalı grupları kullanabilir, ses kanalları arasında oluşabilecek senkronizasyon sorunlarını engelleyebilirler.

ASIO arabirimi, ses donanımının desteklediği sayıda giriş ve çıkış kanalını soyutlayarak ses uygulamalarının kullanımına açar. Ses uygulaması, bu kanallar arasından kullanacağı kadarını etkin hale getirir. Ses verisinin donanım sürücüsüyle uygulama arasında dolaşımı için, her kanala özel bir çift arabellek (*double buffer*) yaratılır. Arabelleklerden biri uygulama tarafından kullanılırken, diğeri de sürücü tarafından kullanılır; bu süreç dönüşümlü olarak devam eder. Ses çıkışı durumunda arabelleklerden birincisi uygulama tarafından doldurulurken, ikincisi ses donanımını tarafından (D/A çevrimi vb. işlemler yapılmak üzere) fiziksel çıkışlara gönderilir. Donanım ikinci arabelleğin sonuna ulaşmadan uygulamanın birinci arabelleği yeni veriyle doldurması gereklidir. Donanım ikinci arabelleğin sonuna ulaştığında, birinci arabellekteki veriyi fiziksel çıkışlara göndermeye başlar; uygulama da ikinci arabelleği veriyle doldurmaya başlar (33).

Arabelleklerin büyüklüğü öncelikle ses donanımı tarafından belirlenir ve ses çıkışı için, ses verisinin uygulamalar tarafından üretilmesiyle ses donanımının fiziksel çıkışlarına ulaşması arasında geçen zamanı (*latency*; gecikme) belirler. Plaka yazılımını çalıştırabilecek bir bilgisayarda, 20 ms ve daha düşük gecikmeyle çalışmak mümkün olmaktadır.

ASIO arabiriminin kullanılmasında, API işlemlerini bir Delphi nesnesine dönüştüren ‘DelphiASIOVST’¹ bileşeni kullanılmıştır. Bu bileşen, ASIO donanım sürücüsü listelenmesi, seçimi, ses akışının denetimi ve arabelleklerin doldurulması gibi işlemleri Delphi *property* (özellik), *method* (yöntem) ve *event*’leri (olay) aracılığıyla gerçekleştirmektedir.

¹ Christian Budde, <http://sourceforge.net/projects/delphiasiovst/>

4.1.2 Çizim Tabletiyle İletişim

Windows işletim sistemlerinde, çizim tabletleriyle iletişim WinTab API'ları aracılığıyla yapılmaktadır. WinTab, çizim tableti üreticilerinin ürünleri için kullandıkları bir standart haline gelmiştir.

WinTab fonksiyonları, sisteme bağlanmış tabletler hakkında çözünürlük ve desteklenen özellikler gibi bilgileri edinmeyi, tablet mesajlarını almak için istekler oluşturmayı ve mesajları sırayla okumayı içermektedir.

Tablet mesajları, işletim sistemi tarafından gönderilen fare mesajlarına benzetilebilir. Mesajlarda, tablet kendi koordinat sistemine göre kalem konumu, basınç seviyesi, kalemin tablet alıcılarının menziline olup olmadığı bilgisi ve kullanım durumuna ve donanıma bağlı olarak diğer ek bilgiler bulunur. Çizim tabletleri sistemde aynı zamanda standart fare aygıtı olarak da tanımlandığı için kalem hareketleri, tablet mesajlarına ek olarak fare mesajları da üretir.

Plaka'da tablet mesajları içinden sadece kalemin yatay ve dikey konumu ile basınç ve menzil durumları işleme alınmaktadır. WinTab fonksiyonları, Centaurix Interactive Designs tarafından üretilmiş 'TTablet' Delphi bileşeni aracılığıyla kullanılmıştır.

4.1.3 Ses Dizileri İçin Scala Dizi Formatı

Plaka'da modellenen çubukların temel frekanslarının belirlenmesi için Scala yazılımının dizi formatı kullanılmıştır. Scala dizi formatı, tınılama (*tuning*) ve tamperaman bilgilerinin kişiler ve uygulamalar arasında aktarılmasında yaygın olarak kullanılan bir dosya formatıdır. Scala yazılımıyla dağıtılan dizi arşivinde 3000'den fazla dosya bulunmaktadır (34).

Scala dizi formatı basit metin tabanlı bir yapıya sahiptir ve dosya uzantısı '.scl'dir. Her dosyada bir dizi tanımı yapılır. Tonlar kesir veya sent cinsinden oranlar olarak tanımlanır. Birinci ton (1/1 oranı) her zaman var kabul edilir ve açıkça belirtilmez. Kesir ve sent cinsinden tanımlar aynı dosya içinde bulunabilir. Her satırın başında bir ton tanımı yapılır. Tanımda nokta işareti bulunuyorsa, tanımın sent cinsinden olduğu var sayılır; yoksa *a/b* biçiminde kesir tanımları yapıldığı kabul edilir (paydası bulunmayan tam sayılar da kullanılabilir). Satırların geri kalanı dikkate alınmaz; dosyayı okuyan kişiler için tonların

tanımının yapılması mümkündür. İlk iki satırdan birincisi, dizi dosyasının genel tanımını, ikinci satır ise dosyadan bulunan ton sayısını belirtir. (Ünlem işaretiyle başlayan satırlar dikkate alınmaz.)

Scala dizi dosyaları, Plaka'da modellenen çubukların tınlama bilgilerinin kaynakları olarak düşünülebilir. Scala dizi formatının uygulanması `ParseScalaFile` (`tabletoscForm.pas, {1}`) yönteminde (*method*) yapılmıştır.

4.2 Çekirdek Nesne Sınıfları

Çekirdek nesne sınıfları (*class*), Plaka'da kullanılan algoritmanın çeşitli bölümlerinin uygulandığı nesnelere tanımlamalarını yapmaktadır.

4.2.1 Filtre Sınıfı

`TWaveguideBPFfilter4p` nesne sınıfı (`WaveguideBandpassFilter.pas`), *banded waveguide* algoritmasında kullanılan *bandpass* filtreyi tanımlar. Filtre, 4. dereceden birer *lowpass* ve *highpass* filtrenin³ arka arkaya kullanıldığı bir yapıya sahiptir. Her iki filtrenin kendi katsayı ve veri alanları (*field*) bulunmaktadır.

`Create` yöntemi {2} işlemcinin SSE komutları için kullandığı *denormal* ayarlarını yapan `SetDAZFTZ` yöntemini {3} çağırır. `SetDAZFTZ` yöntemi bir *assembly* yöntemidir ve işlemcinin geçerli durumuna “SSE komutlarında ortaya çıkan *denormal* sayıları 0'a eşitleme” durumunu ekler. (*Denormal* sayılar hakkında bilgi için bkz. bölüm 4.5) `Initialize` yöntemi {4}, frekans, örnekleme hızı ve *gain* (kazanç) verileri alır; filtrelerin katsayılarını hesaplar. Hesaplanan katsayılar `TWaveguideBPFfilter4p` nesnesinin `InternalState` alanında saklanır. `Initialize` yönteminin çağırılmasından sonra filtre nesnesi kullanıma hazırdır.

Filtrenin işleme yöntemi, `Process` yöntemidir {5}. `Process` yöntemi, `Single` (32 bit uzunluğunda kayan nokta) türünde veri alan ve yine `Single` türünde veri geri döndüren bir fonksiyondur. Kısmen Delphi kodundan, kısmen *assembly* kodundan oluşmaktadır. Filtre

³ <http://www.musicdsp.org/showArchiveComment.php?ArchiveID=181> sayfasında tanımlanan algoritma kullanılmıştır.

algoritmasının bir kısmı, *assembly* kodunda SSE2 komutlarıyla vektörize edilmiş bir halde hesaplanır {6}. SSE2 komutlarının kullanılması, filtre algoritmasını önemli ölçüde hızlandırmaktadır. (*Assembly* kodlarının Delphi denklemleri {7}'de görülebilir). Filtre içindeki bütün hesaplamalar *Double* (64 bit uzunluğunda kayan nokta) tipinde değerler ile yapılmaktadır.

4.2.2 Delay Sınıfları

TWaveguideDelay ve *TWaveguideInterpolatedDelay* nesne sınıfları (*WaveguideDelay.pas*), *banded waveguide* algoritmasında kullanılan *delay* hatlarını tanımlar.

TWaveguideDelay nesne sınıfının *Create* yöntemi {8}, *delay* uzunluğu verisini alır, *delay* arabelleğini (*DelayBuffer*) uzunluğa uygun büyüklükte yaratır ve içini 0 (sessizlik) ile doldurur. *DelayBuffer* sirküler kullanımlı bir arabellektir. Konumu *DelayPosition* alanında saklanır (*DelayPosition* 0'dan başlar).

Process yöntemi {9}, *delay* hattına veri gönderilen ve veri alınan fonksiyondur. *Single* türünde veri alır ve aynı türde veri geri döndürür. *Process* yönteminde öncelikle *DelayBuffer*'ın *DelayPosition* tarafından belirlenen adresinden veri okunarak geri dönüş değişkenine (*Result*) kaydedilir. *SampleIn* değişkeni içinde gelen yeni ses verisi, aynı konuma yazılır. Daha sonra, geri dönüş verisi *DelayFeedback* katsayısıyla çarpılır. Eğer çarpım sonucu *denormal* sayı oluşturuyorsa *DelayBuffer*'daki yeni kaydedilen değere eklenir. Son olarak *DelayPosition* değeri 1 arttırılır; eğer yeni konum *DelayBuffer*'ın uzunluğunu aşıyorsa 0 konumuna (başa) dönlür.

TWaveguideInterpolatedDelay, *TWaveguideDelay* nesne sınıfının ara kestirim (*interpolation*) özellikli halidir. Örnekleme hızının tamsayı çarpanı olmayan frekanslarda (ör. 48000 Hz örnekleme hızında 3500 Hz'lik titreşim modu için), *delay* işleminin geri dönüş değerinin konumu iki örnek (*sample*) arasına düşer. *Delay* uzunluğunun tamsayıya yuvarlanması sonucunda, duyulabilir büyüklükte frekans sapması oluşabilir. Ara kestirim özelliği, iki gerçek örnek konumu arasına düşen 'hayali' örneklerin algoritmik olarak hesaplanmasını mümkün kılarak frekans sapmasını en aza indirmeyi amaçlar. (Bu özellik işlem gereksinimini büyük ölçüde arttırmaktadır.)

Create yöntemi {10}, *delay* hattı uzunluğunu tamsayı yerine *Single* türde alır. *DelayBuffer*'ın uzunluğu bu değerin yukarı doğru yuvarlanması sonucu elde edilen tamsayı olarak belirlenir. *Single* türdeki *delay* uzunluğunun ondalık kısmı *DelayFractionalDifference* alanında saklanır.

Process yönteminde {11}, *TWaveguideDelay*'den farklı olarak, geri döndürülen değerin saptanmasında *Hermite Interpolation* algoritması kullanılır. Geri dönüş verisinin saptanmasında, *DelayPosition*'daki konum doğrudan kullanılmaz; bu konum arada kalan hayalî örneğin hemen öncesinde gelen gerçek örneği belirler. Buna göre, hayalî örneğin öncesinde ve sonrasında gelen ikişer örnek saptanır ve *DelayFractionalDifference* değeri ile birlikte *HermiteInterpolation* yöntemine {12} gönderilir. *HermiteInterpolation* yöntemindeki algoritma⁴, bu dört değerin arasındaki farkların 'eğim'lerinden yola çıkarak arada kalan örneği hesaplar.

4.2.3 Waveguide Sınıfı

TWaveguideThread (*WaveguideOsc.pas*), *TThread* nesne sınıfı kökenli bir tanımlamadır. Bu sınıfa ait her bir nesne, bir çubuk modelini temsil eder. *TThread* nesne sınıfı, *thread*'lerle (iş parçacığı) ilgili Windows API fonksiyonlarını kapsayan VCL nesne sınıfıdır. *TWaveguideThread*'in *TThread* kökenli olması, her çubuk modelinin ayrı bir *thread*'de çalışması anlamına gelmektedir. Bu sayede, birden fazla işlemcili bilgisayarlarda, işletim sistemi otomatik olarak yük dağıtımını yapabilir.

Create yöntemi {13}, *TWaveguideThread*'i durdurulmuş halde yaratarak, nesnenin bazı alanlarının başlangıç değerlerini belirler. Nesnenin çalışmasını başlatmak, nesneyi yaratan koda bırakılır. (Başlatmadan önce bazı alanların doldurulması gereklidir.)

Execute yöntemi {14}, nesne başlatıldığında çalıştırılan yordamdır. Bu yordamın üç aşamadan oluştuğu söylenebilir. Birinci aşamada yerel değişkenler ve bazı nesne alanları kullanıma hazırlanır. Nesneye denk gelen *thread*'e Delphi'nin oluşturduğu *SetName* yordamı {15} kullanılarak bir isim verilir (isim verilmesi hata ayıklama sürecinde yararlı olmaktadır). Daha sonra nesnenin ses çıkışı için kullanacağı çift arabellek

⁴ <http://www.musicdsp.org/showArchiveComment.php?ArchiveID=93> sayfasında tanımlanan algoritma kullanılmıştır.

(`OutDoubleBuffer`) uygun büyüklükte yaratılır ve 0'la doldurulur {16}. Nesne için dışarıdan belirlenen alan değerlerine göre yerel değişkenlerin başlangıç değerleri atandıktan sonra, dışarıdan alınan basınç ve hız değerlerindeki değişimlerin yumuşatılması için kullanılan katsayılar belirlenir {17}. Son olarak *banded waveguide* algoritması için kullanılacak filtre-*delay* zincirlerinin `Modes` ve `ModeGains` listelerinden alınan bilgilere göre yaratıldığı küçük bir döngüye girilir {18}.

`Modes` listesi titreşim modlarının frekans oranlarını `Single` değerler olarak belirler. `ModeGains` listesi de titreşim modlarının genlik oranlarını `Single` değerler olarak belirler. Her iki liste de nesneyi yaratan kod tarafından nesne çalıştırılmadan önce doldurulur. Her titreşim modunun frekansı, `OscFrequency` alanındaki temel frekansa göre hesaplanır. Nyquist frekansını aşıyorsa, döngü durdurulur. Aşmıyorsa, `Delays`, `Filters` ve `ZeroCounters` listelerinin büyüklüğü ve `ModeCount` sayacı 1 artırılır.

`Delays` listesi *delay* nesnelerini saklayan listedir. Ara kestirim özelliği istenmişse (`OscInterpolated`), listenin sonuna bir `TWaveguideInterpolatedDelay` nesnesi, istenmemişse bir `TWaveguideDelay` nesnesi eklenir. *Delay* hatlarının uzunluğu örnekleme hızının titreşim modu frekansına bölümüne eşittir.

`Filters` listesi *bandpass* filtre nesnelerini saklayan listedir. Bir `TWaveguideBPFilter4p` nesnesi `Create` yöntemiyle yaratıldıktan sonra `Initialize` yöntemi frekans, örnekleme hızı ve *gain* değerleriyle çağrılır. (Frekans, titreşim modunun frekansına eşittir.)

`ZeroCounters` listesi filtrelerin kullanılmadığı durumlarda (filtreden sessizlik geçtiğinde) devre dışı bırakılmaları için kullanılan bir listedir. Listedeki her alana 0 başlangıç değeri verilir.

Döngü bittikten sonra `ModeCount` sayacının 0'dan büyük olup olmadığı kontrol edilir; eğer değilse nesneyi yaratan koda bu nesne tarafından ses üretilmeyeceğini bildirmek için `HasNoModes` alanına `True` değeri atanır. Son olarak birinci aşamanın bittiğini bildirmek için `OscReady` alanına `True` değeri atanır ve ikinci aşamaya geçilir.

İkinci aşama, nesneye sonlanma sinyali (`Terminate`) gönderilinceye kadar tekrarlanan bir `While...Do` döngüsünden oluşur {19}. Döngünün içinde çift arabelleğin bir bölümünü dolduracak miktarda veri üreten bir `Repeat...Until` döngüsü bulunmaktadır.

Bu döngü içinde öncelikle dışarıdan alınan `OscPressure` ve `OscVelocity` değerlerindeki değişimlerin yumuşatıldığı basit algoritmalar bulunur {20}.

`OscPressure` alanının değer farklarını yumuşatan algoritmada, yeni değer ile eski değer (`PrevPressure`) arasındaki fark `PressureSmoothingCount` alanında belirtilen sayıda adıma bölünerek `InternalPressure` değişkenine atanır. `OscPressure` değeri her değiştiğinde yumuşatma işleminin hedefi ve adım büyüklüğü güncellenerek devam eder. Bu yumuşatmanın sebebi, çoğu çizim tabletinin basınç verilerinin ses uygulamalarında doğrudan kullanılacak çözünürlükte olmamasıdır.

`OscVelocity` alanındaki değer ise iki yönlü yumuşatmadan geçer. Düz yönde {21}, yeni değer ile eski değer (`PrevVelocity`) arasındaki fark `VelocitySmoothingCount` değerinin yarısı kadar adımda `InternalVelocity` değişkenine atanır. Daha sonra ters yöne geçilir {22}; bu durumda `InternalVelocity`'nin değeri 0'a doğru düşmeye başlar. Eğer `OscVelocity` alanında yeni değişiklikler olmazsa, sonunda `InternalVelocity` 0'a eşit olur. Bu algoritmanın kullanılmasının sebebi, tablet mesajlarından alınan verilerden hareketler anlaşılabilirken, hareketsizliğin doğrudan anlaşılabilir olmasıdır.

Yumuşatma işlemlerinden sonra, çift arabelleğin doldurulması süreci başlar {23}. `OutDoubleBuffer` alanının `BufferToggle` tarafından belirlenen bölümünün `a` değişkeni tarafından belirlenen konumuna 0 değeri atanır. (Titreşim modları için hesaplanan veriler bu konuma eklenecektir.) Ses kaynağından Delphi'nin `Random` fonksiyonu ile elde edilen veri, `InternalPressure` ve `InternalVelocity` değerleriyle çarpılır. Ek olarak, kalemin dikey konumuna ve çubuğun uç durumlarına bağlı olarak gerçekleşen sönümlenme

$$1 - \frac{(\text{Abs}(\text{OscPressurePoint} - \text{OscVerticalDampingCenter}) * \text{OscVerticalDampingAmount})}{\text{OscVerticalDampingAmount}}$$

işlemiyle hesaplanarak ses kaynağı verisiyle çarpılır ve sonuç `u` değişkenine atanır. Ses kaynağından alınan verinin genliğinin 0'ı (aşağı ya da yukarı doğru) aşması için, `InternalPressure` ve `InternalVelocity` değerlerinin ikisinin de 0'dan büyük olması gerekir. `OscPressurePoint`'e bağlı yapılan işlemin etkisi, `OscVerticalDampingAmount` değerine bağlı olarak sınırlandırılır.

Ses kaynağından alınan verinin titreşim modlarının *waveguide* zincirlerinden geçirilmesi için bir `For` döngüsü bulunmaktadır {24}. Bu döngüde öncelikle ses kaynağından alınan veri, söz konusu titreşim modunun genlik katsayısıyla çarpılır. Daha sonra, ses kaynağından alınan veri 0'a eşitse ve `ZeroCounters` listesinde bu mod için saklanan veri 1000'den küçükse veya ses kaynağından alınan veri 0'ı aşıyorsa, *bandpass* filtreden geçer. Bu sayede 1000 örnekten uzun sessizlik durumlarında filtre devre dışı kalarak işlem hacmi azaltılır.

Filtreden sonra, titreşim modu için kullanılan *delay* nesnesinin `Feedback` alanı `OscFeedback`, `InternalPressure` ve `OscPressurePoint` değerlerine bağlı olarak güncellenir. `InternalPressure` değerinin etkisi, değer %1'i 1'den çıkartılarak, `OscPressurePoint`'in etkisi de

```
1 - (Abs(OscPressurePoint - OscVerticalDampingCenter) *  
OscVerticalDampingAmount * 0.01 * InternalPressure)
```

işlemiyle hesaplanır. Bu `Feedback` hesaplaması, basınç ve uç sönümlenmesine bağlı olarak çubuk içindeki sönümlenmeyi ayarlar. Son olarak ses kaynağından gelen veri, *delay* nesnesinden geçirilir ve alınan sonuç `OutDoubleBuffer`'a eklenir.

`OscVelocity` değişikliklerinin yumuşatıldığı algoritmanın ikinci bölümünden sonra, `a` değişkeni bir artırılır ve `Repeat...Until` döngüsünün sonuna gelinir. Döngü, `OscOutBufferSize` alanında belirtilen sayıda tekrar edildikten sonra, çift arabelleğin ilgili bölümü doldurulmuş olur. Tekrar `Terminated` alanının değeri kontrol edilir; eğer `True` ise ikinci aşamanın `While...Do` döngüsü durdurularak üçüncü aşamaya geçilir. `False` ise, `TWaveguideThread` nesnesi `Suspend` yöntemiyle duraklatılır {25}. Çift arabelleğin diğer bölümünün doldurulması gerektiğinde, nesneyi yaratan kod `Resume` yöntemiyle nesneyi devam ettirmelidir.

Üçüncü aşamada {26}, nesnenin yok edilmesinden önce *banded waveguide* algoritması için yaratılan filtre ve *delay* nesneleri yok edilerek kullandıkları bellek alanları boşaltılır.

4.3 Ses Çıkışı

Ses çıkış mekanizması `TThread` kökenli `TAudioEngine` (`AudioEngine.pas`) nesne sınıfı tarafından sağlanmaktadır. Bu sınıftan sadece bir nesne yaratılmaktadır. Bu nesne, kendi *thread*'inde çalışır. Ses akışı için zamanlamanın önemli olması nedeniyle, ilgili işlemler, uygulama arayüzünün işlemleri tarafından engellenmeyecek şekilde yürütülmelidir. Nesne *thread*'inin öncelik seviyesi (*priority*) ayarlanarak bu durumun sağlanması mümkündür. Öncelik seviyesi, nesneyi yaratan kod tarafından `Priority` özelliğine önem değerinin atanmasıyla belirlenir.

`Create` yöntemi {27}, öncelikle ASIO işlemleri için kullanılacak `TAsioHost` sınıfından bir nesne (`AsioHost`) yaratır. `AsioHost`, olay yordamları atandıktan sonra kullanıma hazırdır. Bundan sonra `TObjectList` sınıfından `OscThreads` nesnesi yaratılır. Bu nesne, kullanımda olan `TWaveguideThread` nesnelerinin listesini tutar. Son olarak, `TAudioEngine` nesnesinin alanlarından bazılarının başlangıç değerleri atanır.

`Destroy` yöntemi {28}, nesne yok edilmeden önce çağrılan yordamdır. ASIO ses akışını durdurur ve `OscThreads` listesindeki `TWaveguideThread` nesnelerini sonlandırır.

`Start` yöntemi {29}, ses akışını başlatmak için; `Stop` yöntemi {30}, ses akışını geçici olarak durdurmak için kullanılır.

`AsioBufferSwitch` yordamı {31}, `AsioHost` nesnesi tarafından ASIO arabelleklerinin doldurulması için çağrılan yordamdır. Arabelleği dolduracak kadar veri üreten bir `For` döngüsünden ve arabellek kullanımını yöneten kodlardan oluşur. `For` döngüsü içinde, tüm `TWaveguideThread` nesnelere veri alan başka bir `For` döngüsü bulunur. Her `TWaveguideThread` nesnesinden alınan veri, eğer stereo ayrışımı yapılıyorsa (`StereoSeparation`), nesnenin dizi içindeki sıra numarasına göre sol ve sağ kanal için hesaplanan katsayılarla çarpılarak sol kanal için `L` ve sağ kanal için `R` değişkenlerine eklenir. Son olarak `L` ve `R`, `MasterVolume` değeriyle çarpılarak ilgili ASIO arabelleklerine yazılır.

4.4 Yazılım Arayüzü

Yazılımın arayüz tasarımında, sadelik, anlaşılabilirlik ve kullanılabilirlik ön planda tutulmuştur. Arayüzdeki bütün fonksiyonların sürekli gerekli olmayacağı varsayımından yola çıkılarak, çeşitli bölümleri isteğe göre ortadan kaldırılabilen bir arayüz mekanizması kullanılmıştır.

Arayüz, beş ana bölümden oluşmaktadır. Üst bölümde yazılımın temel fonksiyonlarını gösteren araç çubuğu, sol bölümde sentezleme parametrelerinin değiştirilebildiği panel, sağ bölümde açılan Scala dizi dosyasının içeriğini gösteren panel, altta yazılımla ilgili bazı anlık bilgileri gösteren durum çubuğu bulunmaktadır. Bu dört bölümün ortasını kaplayan beşinci bölüm, ‘çalma’ etkinliğinin gerçekleştiği temsilî çubukları gösteren bölümdür.

4.4.1 Araç Çubuğu

Araç çubuğu, Plaka'nın temel fonksiyonlarını ikonlar halinde gösteren bir panelden oluşur. ‘Dizi Aç’ düğmesi, yeni bir dizi yüklemek için kullanılır. ‘Parametreler’ ve ‘Dizi İçeriği’ düğmeleri, sırasıyla sol ve sağ panelleri göstermek veya gizlemek için kullanılır. ‘Ayarlar’ düğmesi, yazılım ile ilgili bazı ayarların değiştirilebileceği bir pencereyi göstermek için kullanılır. ‘Yeniden Başlat’ düğmesi, işlem yoğunluğu nedeniyle veya kullanıcının isteği üzerine ses akışı durdurulduğunda görünür hale gelir ve ses akışını devam ettirmek için kullanılır. Bunlara ek olarak, araç çubuğunun en sağında, genel ses seviyesini gösteren iki çubuk bulunur. Bu çubuklar 50 ms’de bir ASIO arabelleklerine gönderilen veriden edinilen genel ses seviyesi bilgisini göstermek üzere güncellenir. Çubukların üstünde kırılma (*clipping*) uyarı göstergeleri bulunmaktadır. Bu göstergeler, ses seviyesi kırılmaya yaklaşıncaya yanar. Araç çubuğu F9 kısayol tuşuyla gizlenebilir.

4.4.2 Parametre Paneli

Parametre paneli, ses dizisi ve sentezleme parametrelerinin değiştirilebildiği bölümdür. Kendi içinde iki bölümden oluşur. Üst bölümde, dizinin başlangıç frekansı, yukarı ve aşağı doğru tekrarları ile, modellenen nesnenin titreşim modu sayısı, nesne türü ve uç durumlarının ayarlanabileceği kontroller bulunur.

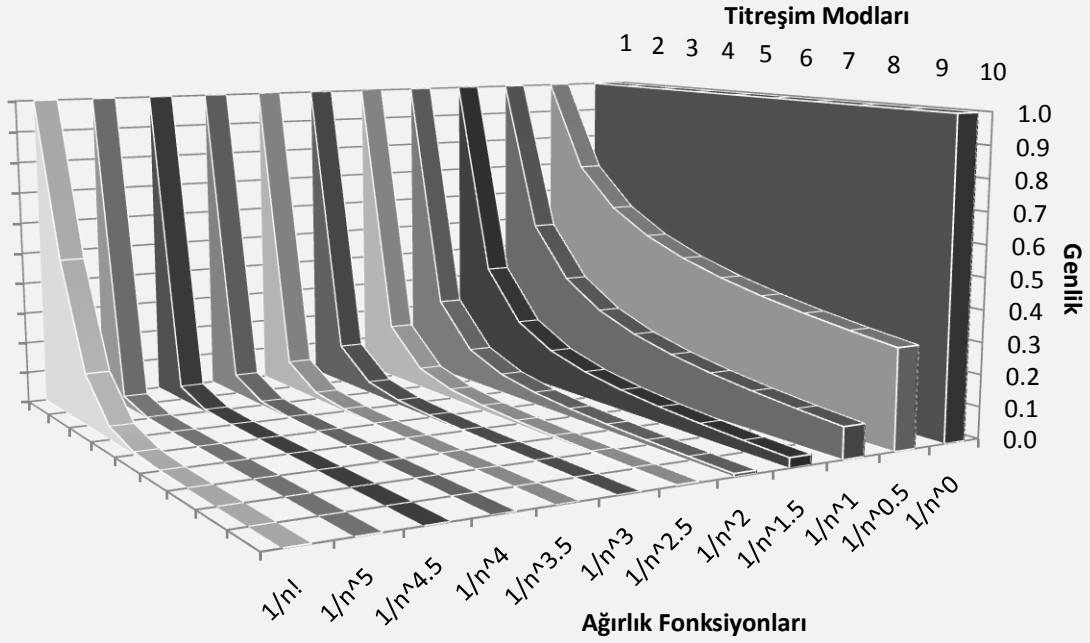
Titreşim modu sayısı hem işlem hacmini kontrol altında tutmak, hem de tınıyı değiştirmek için kullanılır. Uç durumları, çubuk modellerinin kalemin dikey konumuna nasıl tepki vereceğini belirler. Nesne türü, algoritmik olarak modları hesaplanan tekbiçim çubuk olabileceği gibi, bir dosyadan yüklenen frekans oranları ve *gain* değerlerine göre de belirlenebilir. Söz konusu dosya formatı, metin bazlı, her satırında birbirinden virgülle ayrılmış frekans oranı ve *gain* değerinden oluşan bir titreşim modunun tanımlandığı bir yapıya sahiptir. Dosya uzantısı '.vmod'dur. Yazılımla birlikte kullanılması için bazı örnek titreşim modu dosyaları hazırlanmıştır. Bu dosyaların içerikleri Tablo 4.1, Tablo 4.2, Tablo 4.3 ve Tablo 4.4'te görülebilir. (Mod tanımları STK⁵ kaynak kodundan ve (35)'ten alınmıştır; Plaka'daki algoritma bu nesnelere bazıları için en uygun algoritma olmayabilir.)

Nesne türü tekbiçim çubuk olduğunda, mod ağırlıklarının (*gain* değerlerinin) ayarlanabileceği bir kontrol kullanılabilir duruma geçer. Ağırlıkların hesaplanması $1/n^a$ formülüyle yapılır. n titreşim modunun sıra numarasını belirtir. a değeri 0.5'lik adımlarla 0'dan 5'e kadar herhangi bir değer olabilir. Buna ek olarak $1/n!$ formülünü kullanmak da mümkündür. Mod ağırlıkları, üretilen sesin tınısını belirlemede önemli bir rol oynar. Mümkün olan ağırlık durumlarının görsel temsilleri Şekil 4.1'de görülebilir. (Üslü sayılar n^a şeklinde gösterilmiştir.)

Bu noktaya kadar olan parametrelerdeki değişikliklerin geçerli olması için, 'Uygula' düğmesine tıklanması gerekir. Alt bölümdeki iki parametrede değişiklik yapıldığında, bu değişiklikler anında etkili olur. Bu iki parametreden biri, genel rezonans katsayısını belirler. Kontrol yukarı sürüklendikçe rezonans artar. Kontrolün 0 ile 200 arasında bir değeri olabilir. Rezonansın hesaplanması $0.9999^{\text{RezonansSeviyesi}}$ işlemiyle yapılır. Diğer parametre, genel ses seviyesini kontrol eder. Bu kontrolde de 201 seviye bulunmaktadır. 20 seviyesi ses seviyesinin olduğu gibi bırakıldığı noktadır; üstüne çıkıldıkça ses seviyesi $1.1^{21-\text{SesSeviyesi}}$ işlemine bağlı olarak artar; altına inildikçe $0.99^{\text{SesSeviyesi}-20}$ işlemine bağlı olarak azalır. Kontrol seviyesi 201'e eşit olduğunda, ses seviyesi 0'a eşitlenir. Parametre paneli, F3 kısayol tuşuyla gösterilebilir.

⁵ Synthesis Toolkit in C++, <http://ccrma.stanford.edu/software/stk/>

Titreşim Modları İçin Ağırlık Fonksiyonlarının Spektral Karakterleri



Şekil 4.1 Titreşim modlarının ağırlık fonksiyonları ve spektral karakterleri

Titreşim Modu	Temel Frekansa Oranı (:1)	Titreşim Modu	Temel Frekansa Oranı (:1)
1	1	4	6.63
2	2.32	5	9.38
3	4.25		

Tablo 4.1 "Armonika.vmod" dosyası için titreşim modlarının frekans oranları

Titreşim Modu	Temel Frekansa Oranı (:1)	Titreşim Modu	Temel Frekansa Oranı (:1)
1	0.996	7	9.016
2	1.004	8	12.833
3	2.979	9	12.807
4	2.993	10	17.281
5	5.705	11	21.976
6	8.998		

Tablo 4.2 "Tibet Çanağı.vmod" dosyası için titreşim modlarının frekans oranları

Titreşim Modu	Temel Frekansa Oranı (:1)	Titreşim Modu	Temel Frekansa Oranı (:1)
1	1	3	10.718
2	4.020	4	18.070

Tablo 4.3 "Akordlu Çubuk.vmod" dosyası için titreşim modlarının frekans oranları

Titreşim Modu	Temel Frekansa Oranı (:1)	Titreşim Modu	Temel Frekansa Oranı (:1)
1	1	5	2.318
2	1.159	6	3.500
3	1.286	7	3.897
4	2	8	4.648

Tablo 4.4 "Teneke Kutu.vmod" dosyası için titreşim modlarının frekans oranları

4.4.3 Dizi İçeriği Paneli ve Durum Çubuğu

Dizi içeriği panelindeki metin kutusu, her yeni dizi açıldığında dizi dosyasının içeriğiyle güncellenir. F4 kısayol tuşuyla gösterilebilir.

Durum çubuğu, 5 bölümden oluşur. Soldan sağa sırayla, kullanılan çizim tabletinin markasını, kalem basınç seviyesini, yüklenmiş olan dizinin tanımını, kullanılan nesne modelini ve ses akışının durumunu gösterir. F10 tuşuyla gizlenebilir.

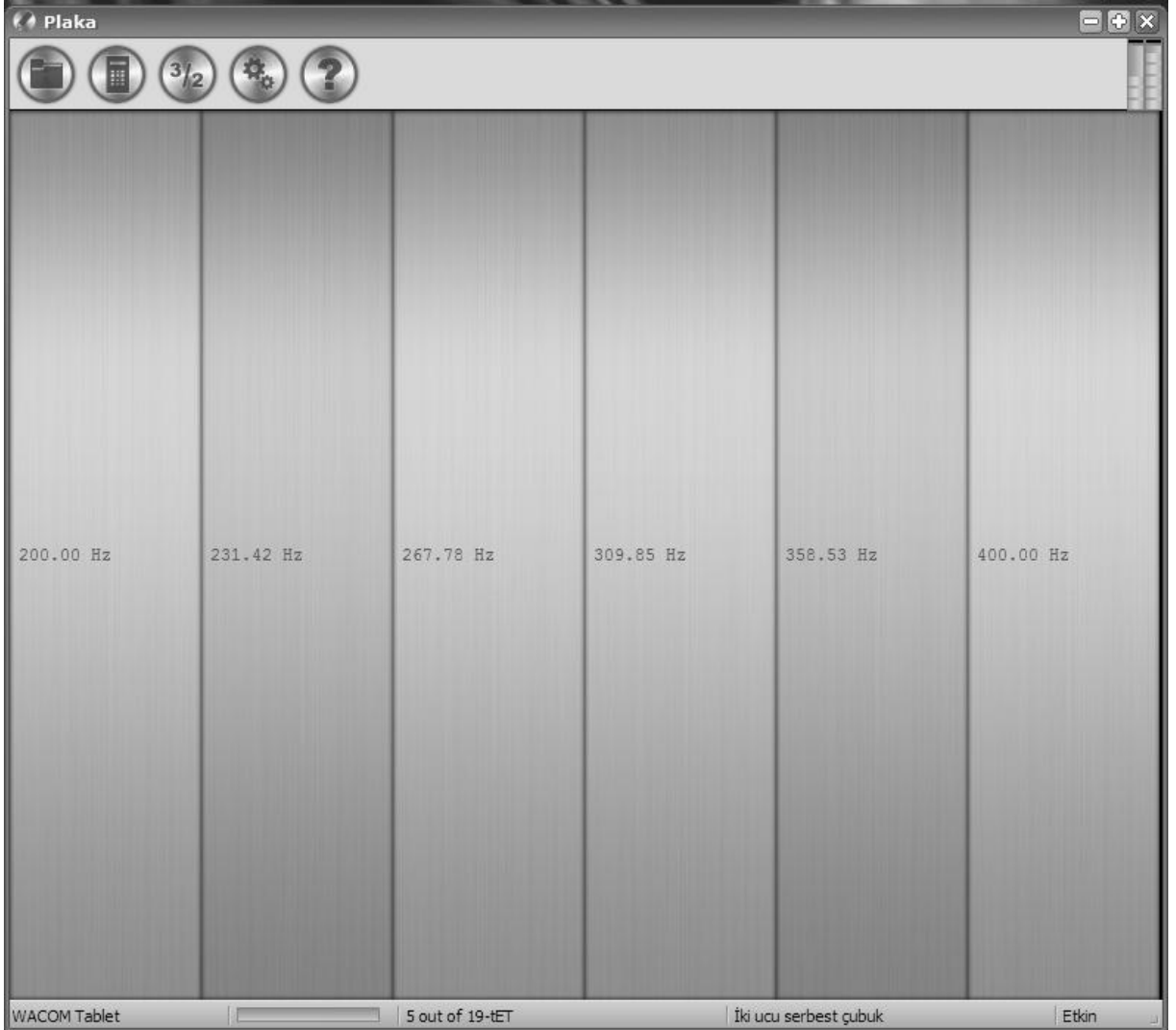
4.4.4 İcra Alanı

Arayüzde modellenen çubukların gösterildiği orta bölüm, 'icra'nın gerçekleştiği bölümdür ve TPaintBox sınıfından ToneZone (tabletoscForm.pas) nesnesi tarafından temsil edilmektedir. Bu bölüm, diğer arayüz bölümlerinden arta kalan bütün alanı kaplar.

Bir dizi dosyası kullanıcı tarafından açıldığında, dizideki seslerin frekans oranları hesaplandıktan (ParseScalaFile {32} ve CalculateRepeats {33} yordamları) ve gerekli TWaveguideThread nesnelere oluşturulduktan (RefreshTones yordamı {34}) sonra, RepaintToneZone {35} yordamının çağrılmasıyla ToneZone nesnesine dizide mevcut olan sesleri temsil eden çubuklar çizilir. Bu çubuklar, ToneZone ile tanımlanan alanın tamamının dikdörtgen parçalara bölünmesiyle oluşur. RepaintToneZone yordamı,

ToneZone nesnesinin OnPaint olayında da çağrılır. Bu sayede nesnenin boyutları değiştiğinde ya da başka bir nedenle yenilenmesi gerektiğinde, çubuklar yeniden çizilir.

Çizim işleminin bazı yönleri kullanıcının tercihlerine göre belirlenir. Bu tercihlerden biri, çubuklar üzerinde temel frekansların gösterilmesidir. Bir diğeri, çubukların tek satır yerine iki satıra bölünmüş olarak çizilmesidir. Bu özellik, perde sayısı yüksek olan dizilerin kullanımında, çubukların sıkışarak fazla ince hale gelmelerini önlemeye yarar. Sonuncu tercih, çubukların düz renkli, gri, basit dikdörtgenler veya temsili, renkli çubuklar olarak çizilmesini belirler.



Şekil 4.2 Yazılım arayüzünde temsili çubuklar

ToneZone nesnesi, çubuk modellerinin görsel temsilinin oluşturulmasına ek olarak, çizim tabletinden alınan verilerin hangi çubuk modeline gönderilmesi gerektiğini belirleyen mekanizmayı da sağlar. Bu mekanizma, nesnenin OnMouseMove olayında gerçekleşir. Bu olaya atanmış yordamda {36}, fare imlecinin konumundan kalemin hangi çubuk üzerinde olduğu saptanır ve ToneIndex değişkenine atanır. Tablettten gelen paketlerin işleme alındığı yordamda da {37}, veriler bu değişkene göre hedef çubuk modeline yollanır.

Bunlara ek olarak, çizim tabletinin bulunmadığı ya da kullanıcı tarafından devre dışı bırakıldığı durumlarda, nesnenin fare düğmelerinin basılma ve bırakılma olaylarında, çubuk modellerine basınç verisi gönderilir. Burada uygulanan ‘hayali’ basınç sabittir ve %50 değerindedir.

4.5 Performans İyileştirmeleri

Eş zamanlı kullanılmak üzere tasarlanan bir ses uygulamasında, eş zamanlı kullanımın sağlanabilmesi için bütün işlemlerin mümkün olduğunca sadeleştirilmesi ve çalışılan donanımın özelliklerine göre en uygun düzene sokulması büyük önem taşır. Yazılım performansı üzerinde en büyük etkiyi yaratan kod bölümleri, sık çalıştırılan işlemlerin bulunduğu küçük kod parçacıklarıdır.

Bu kod parçacıklarının büyük çoğunluğunda kayan noktalı (*floating point*) sayılarla işlem yapılmaktadır. Intel Pentium 4 sınıfı işlemcilerde bölme işlemlerinin, çarpma işlemlerinden 6 kata kadar daha uzun sürmesi nedeniyle (36), en az bir sabit sayıyı içeren kayan noktalı bölme işlemleri, sabit sayının tersinin kullanıldığı çarpımlara dönüştürülmüştür.

Kayan noktalı işlemlerle ilgili bir diğer önemli nokta, *denormal* sayılardır. *Denormal* sayılar, 0’a çok yakın olan, işlemin yapıldığı kayan noktalı sayı formatında normal hassaslıkta ifade edilemeyecek kadar küçük sayılardır (37). Bu sayıların kayan noktalı işlemlerde kullanılması, ciddi performans kaybına yol açar. Bu durumdan kaçınmak için, TWaveguideDelay ve TWaveguideBPFILTER4p nesnelerinin Process yöntemlerinde, *denormal* sayıların oluşma ihtimali bulunan aritmetik işlemlerinin sonuçları kontrol edilmekte, *denormal* sayıyla karşılaşıldığında, bu sayının sonraki işlemlere geçmemesi için, sayıyı tutan değişkenin değeri 0’a eşitlenmektedir. Buna ek olarak, SSE komutları için *denormal* sayıların işlemci tarafından 0’a eşitlendiği durum kullanılmaktadır.

Aritmetik işlemlerini hızlandırmanın bir başka yolu da, sonuçları birbirine bağlı olmayan aynı türden işlemleri, vektörel işlemlere dönüştürüp SSE komutları kullanarak programlamaktır. SSE komutları `TWaveguideBPFILTER4p` nesnesinin `Process` yönteminde, filtre verilerinin filtre katsayılarıyla çarpılmasında kullanılmıştır.

Yazılım performansı için bir başka önemli nokta, kod bölümlerinin fonksiyonlarına göre birbirinden ayrılması ve uygun öncelik seviyelerinde çalıştırılmasıdır. Eş zamanlı bir ses uygulamasında, arayüz ile ilgili işlemlerde küçük gecikmeler hoş görülebilir, fakat ses akışında kesintiler kabul edilemez. Dolayısıyla arayüz işlemlerinin, ses akışındaki işlemlere engel olmaması gerekir. Bu durum, *banded waveguide* algoritmalarının ve ASIO ses akışını sağlayan kodların arayüzden ayrı *thread*'lerde çalıştırılmasıyla sağlanmaktadır. Bu *thread*'lerin öncelik seviyeleri en yüksek seviye olarak ayarlanmıştır. Böylelikle işlem yükü işlemcinin sınırlarına yaklaştığında, arayüz tepkilerinde gecikme oluşmasına karşın ses akışının kesintisiz devam etmesi sağlanmaktadır.

5. SONUÇLAR

Bu çalışmada, fiziksel modelleme sentezleyicilerinin etkin biçimde kullanılmak için sentezleme tekniğine uygun kontrol mekanizmalarına ihtiyaç duyduğu; arayüzden yola çıkarak sentezleme tekniği tasarlanmasının olumlu sonuçlar yaratabileceği; arayüz ile sentezleyici eşleşmelerinde MIDI'nin temel alınması yerine, uygulamaya özel iletişim ve kullanım sistemlerinin tasarlanması gerektiği ve çizim tabletlerinin ses uygulamalarında icra arayüzü olarak kullanımının iyi sonuçlar verebileceği fikirlerinden yola çıkılarak, çizim tabletlerine yönelik, fiziksel modelleme temelli, Plaka adı verilen bir sentezleme yazılımı geliştirilmiştir. Geliştirilen yazılımın kullanılmasından edinilen deneyimlerle, sözü edilen fikirler hakkında bazı genel sonuçlara varılabilir.

Kullanılan fiziksel modelleme temelli sentezleme algoritmasının uygun bir kontrol mekanizmasıyla eşleştirilmesi suretiyle, algoritmanın canlı icrada kullanılabilirliği büyük ölçüde sağlanmıştır. Kontrol mekanizması sayesinde, çubuk modelinin bütün olanakları kullanılarak, modelden basit ve tutarlı biçimde, tını paleti dahilinde çeşitli tepkilerin alınması mümkün olmuştur.

Buna bağlı olarak, fiziksel modelleme temelli bir sentezleme tekniğinin tasarlanmasında, var olan bir arayüz aygıtından yola çıkmanın diğer yaklaşımlara makul bir alternatif oluşturduğu söylenebilir. Bu tür bir tasarımla, hem kontrol mekanizmasının hem de sentezleme tekniğinin bütün olanaklarından faydalanılmıştır.

Söz konusu olanaklardan faydalanılmasında, uygulamaya özel bir işleyişin kullanılmasının rolü önemlidir. Kontrol mekanizmasının sağladığı hassasiyet, çubuk modelinin etkin bir şekilde çalınmasında merkezî bir konuma sahiptir. Çizim tabletinin kendine özel iletişim yöntemi ve buna göre tasarlanmış bir yazılımla, tabletin özelliklerinden tam anlamıyla yararlanmak mümkün olmuştur.

Buna rağmen, çizim tabletinin, böyle bir uygulamada yetersiz kalan yanları olmuştur. Basınç ve hareket çözünürlüğü çoğu MIDI kontrol aygıtından daha hassas olsa da, değerlerdeki değişiklik adımları büyük bulunmuş, yumuşatma algoritmalarının kullanılması gerekmiştir. (Benzer yumuşatma algoritmalarının MIDI kontrol aygıtları için de kullanılması kaçınılmazdır.)

Çizim tabletinin en önemli yetersizliği, kontrol mekanizmasının doğasına bağlanabilir. Bu kontrol mekanizması ve tasarlanan yazılım arayüzü ile sadece bir çubuk modelinin titreştirilmesi mümkündür. (Birden fazla çubuğun titreşimi birbirinden bağımsız olarak tamamen sönümlenene kadar devam edebilir.) Bu eksikliğin giderilmesi, aynı anda birden fazla kalem kullanabilen tabletler ve benzer kontrol aygıtlarıyla ve yazılımın bu aygıtlarla çalışacak hale getirilmesiyle sağlanabilir. Buna ek olarak, tablet mekaniğinin ve el ile göz arasındaki koordinasyonun doğası gereği, çubuklar arasında hızlı geçiş yapmak kolay değildir. Monitörlere entegre olmuş çizim tabletlerinin veya 'tablet PC'lerin kullanımı, geçiş zorluğunu önemli ölçüde azaltabilir. Bunun yanı sıra, yazılım arayüzündeki temsili çubukların tasarlanandan farklı düzenlerde arayüze yerleştirilmesi ve aynı anda birden fazla kalemle çalışan tabletlerin kullanımı bu konuda kolaylık sağlayabilir.

Çizim tabletiyle birlikte kullanılan kalemin de, bu projedeki uygulama türünde sorun yaratma ihtimali bulunmaktadır. Görece daha hafif kuvvetlerle kullanılmak üzere tasarlanan tablet kalemlerinin bu tür ses uygulamalarında kullanılması, kullanım sürelerinin kısalmasına neden olabilir. (Aynı ihtimalin çizim yüzeyi için de geçerli olduğu düşünülmektedir.)

Yine de ses uygulamalarında çizim tableti, kullanılması kolay bir mekanizmadır. Çizim tableti ve Plaka yazılımı birlikte bir enstrüman olarak ele alınırsa, çalmayı öğrenmenin oldukça kolay olduğu ve çok çabuk sonuç verdiği görülebilir. Bu yanı sıra, Plaka ve benzer yazılımların pedagojik uygulamalar ve müzik terapisi gibi alanlarda da kullanılabileceği düşünülmektedir.

KAYNAKLAR

1. **Paradiso, Joseph A. ve O’Modhrain, Sile.** Current Trends in Electronic Music Interfaces. *Journal of New Music Research*. 2003, Cilt 32, 4.
2. Lemur Overview. *JazzMutant*. [Çevrimiçi] *JazzMutant*. [Alıntı Tarihi: 24 Nisan 2008.] http://www.jazzmutant.com/lemur_overview.php.
3. **Baum, Robert Thomas, Curry, James Edward ve Winter, Jeffrey Ian.** *RF-based dynamic remote control for audio effects devices or the like. Patent # 20070175321 A.B.D.*, 2 Ağustos 2007.
4. **Knapp, R. Benjamin ve Lusted, Hugh S.** A Bioelectric Controller for Computer Music Applications. *Computer Music Journal*. 1990, Cilt 14, 1, s. 42-47.
5. **MIDI Manufacturers Association.** MIDI Message Table 1. *MIDI Manufacturers Association Web Site*. [Çevrimiçi] [Alıntı Tarihi: 27 Nisan 2008.] <http://www.midi.org/techspecs/midimessages.php>.
6. **Knauer, Christian.** How the Wacom cordless, batteryless pen works. *Wacom Europe GmbH*. [Çevrimiçi] [Alıntı Tarihi: 25 Nisan 2008.] http://www.wacom-europe.com/_bib_user/downloads/tech_i3_en.pdf.
7. *Is the Tablet Pen an Ergonomic Alternative to the Computer Mouse?* **Sorgatz, Hardo, et al.** Stockholm, 2007. Proceedings of the 8th International Conference on Work With Computing Systems.
8. **Fournel, Nicolas.** Wmidi. *Nicolasfournel.com*. [Çevrimiçi] [Alıntı Tarihi: 25 Nisan 2008.] <http://www.nicolasfournel.com/wmidi.htm>.
9. **Carstens, Jacob.** Products - Tablet 2 MIDI. *Livelab.dk*. [Çevrimiçi] [Alıntı Tarihi: 25 Nisan 2008.] <http://www.livelab.dk/tablet2midi.php>.
10. *Toward a generalized friction controller: from the bowed string to unusual musical instruments.* **Serafin, Stefania ve Young, Diana.** Hamamatsu, 2004. Proceedings of the 2004 Conference on New Interfaces for Musical Expression.

11. *The Electronic Sitar Controller*. **Kapur, Ajay, et al.** Hamamatsu, 2004. Proceedings of the 2004 Conference on New Interfaces for Musical Expression.
12. *AoBachi: A New Interface for Japanese Drumming*. **Young, Diana ve Fujinaga, Ichiro.** Hamamatsu, 2004. Proceedings of the 2004 Conference on New Interfaces for Musical Expression.
13. **Blades, James ve Holland, James.** Vibraphone. *The New Grove Dictionary of Music and Musicians*. Londra : Macmillan Publishers Limited, 2002. Cilt 26.
14. **Tuck, Kevin.** A Teacher's Guide to Mallet Percussion Instruments. *KT Percussion*. [Çevrimiçi] [Alıntı Tarihi: 9 Mayıs 2008.]
<http://www.ktpercussion.com/malletinstruments.html>.
15. **King, Alec Hyatt.** Musical glasses. *The New Grove Dictionary of Music and Musicians*. Londra : Macmillan Publishers Limited, 2002. Cilt 17.
16. **The Houston Museum of Natural Science.** BFranklin Press Images. *The Houston Museum of Natural Science Web Site*. [Çevrimiçi] [Alıntı Tarihi: 9 Mayıs 2008.]
http://www.hmns.org/generic/bfranklin_press_images.asp?r=1.
17. **Shostak, Dean.** About Benjamin Franklin's Glass armonica. *Crystal Concert*. [Çevrimiçi] [Alıntı Tarihi: 9 Mayıs 2008.] http://www.crystalconcert.com/optional/glass_armonica.htm.
18. *HyperPuja: A Tibetan Singing Bowl Controller*. **Young, Diana ve Essl, Georg.** Montreal, 2003. Proceedings of the 2003 Conference on New Interfaces for Musical Expression.
19. **Tuner, Dyane, Dickerson, Robert ve Dickerson, June.** Tibetan Bowls. *Reiki Sounds*. [Çevrimiçi] [Alıntı Tarihi: 9 Mayıs 2008.]
<http://homepage.ntlworld.com/reiki.sounds/TIBETAN%20BOWLS%20&%20TUNING%20FORKS.htm>.
20. **Smith, Julius O.** Physical Modeling using Digital Waveguides. *Computer Music Journal*. 1992, Cilt 16, 4.
21. **Carrillo, Alfonso Antonio Pérez.** *Gesture based synthesis of bowed string instruments*. Barcelona : Universitat Pompeu Fabra Teknoloji Bölümü Doktora Tezi, 2006.

22. *Modeling and Control of Performance Expression in Digital Waveguide Models of Woodwind Instruments*. **Scavone, Gary P.** Hong Kong, 1996. Proceedings of the 1996 International Computer Music Conference.
23. *Working with Banded Waveguides and Friction in Musical Contexts*. **Reyes, Juan.** Taipei, 2006. Proceedings of the 2006 International Workshop on Computer Music and Audio Technology.
24. **Akay, Adnan.** Acoustics of friction. *The Journal of the Acoustical Society of America*. 2002, Cilt 111, 4.
25. **Essl, Georg ve Cook, Perry R.** Measurements and efficient simulations of bowed bars. *The Journal of the Acoustical Society of America*. 2000, Cilt 108, 1.
26. **Serafin, Stefania.** *The Sound of Friction: Real-time Models, Playability and Musical Applications*. Stanford : Stanford University Müzik Bölümü Doktora Tezi, 2004.
27. **Fletcher, Neville H. ve Rossing, Thomas D.** *The Physics of Musical Instruments*. New York : Springer-Verlag, 1998.
28. **Ullmann, D.** Life and work of E.F.F. Chladni. *The European Physical Journal Special Topics*. 2007, Cilt 145, 1.
29. **Sigma Xi, The Scientific Research Society.** Figure 1. Chladni Patterns. *American Scientist Online*. [Çevrimiçi] [Alıntı Tarihi: 10 Mayıs 2008.]
<http://www.americanscientist.org/template/AssetDetailNoFrame;jsessionid=baa9...?assetId=27365>.
30. *Vibration Modes of a C4 Handbell: Holographic Interferometry and Finite Element Analysis*. **Fisher, Matthew J.** Lexington, 2001. Proceedings of The National Conference On Undergraduate Research (NCUR) 2001.
31. *Banded Waveguides: Towards Physical Modeling of Bowed Bar Percussion Instruments*. **Essl, Georg ve Cook, Perry R.** Beijing, 1999. Proceedings of the International Computer Music Conference.
32. **Brancheriau, Loïc, Baillères, Henri ve Sales, Christian.** Acoustic resonance of xylophone bars: experimental and analytic approaches of frequency shift phenomenon during the tuning operation of xylophone bars. *Wood Science and Technology*. 2005, Cilt 40, 2.

33. **Scheffler, Stefan.** ASIO SDK 2.2. *Steinberg Media Technologies GmbH.* [Çevrimiçi] [Alıntı Tarihi: 17 Mayıs 2008.] <http://www.steinberg.net/329+M52087573ab0.html>.
34. **de Coul, Manuel Op.** Scala Home Page. *Huygens-Fokker Foundation, centre for microtonal music.* [Çevrimiçi] [Alıntı Tarihi: 17 Mayıs 2008.] <http://www.huygens-fokker.org/scala/>.
35. *Spectral Characteristics of the Musical Iced Tea Can.* **Sturm, Bob L. ve Pope, Stephen T.** Miami, 2004. Proceedings of the 2004 International Computer Music Conference.
36. **Intel Corporation.** Intel 64 and IA-32 Architectures Software Developer's Manuals Voluem 2A: Instruction Set Reference A-M. *Intel Web Site.* [Çevrimiçi] [Alıntı Tarihi: 22 Mayıs 2008.] <http://download.intel.com/design/processor/manuals/253666.pdf>.
37. **de Soras, Laurent.** Denormal numbers in floating point signal processing applications. *Music-DSP Source Code Archive.* [Çevrimiçi] [Alıntı Tarihi: 22 Mayıs 2008.] <http://www.musicdsp.org/files/denormal.pdf>.

EK-1: YAZILIM KAYNAK KODU

Proje kitabının arka kapağının iç yüzündeki CD’de, projede anlatılan Plaka yazılımı ve yazılımın kaynak kodu bulunmaktadır. Kaynak kodunu oluşturan dosyalar, Delphi 7’nin proje formatındadır. Buna ek olarak, projede kullanılan dış kaynaklı bileşenler de CD’ye eklenmiştir. CD’nin klasör yapısı aşağıdaki gibidir:

❖ \Ekstra

Dış kaynaklı dosyalar.

▪ \Asio4All

ASIO özelliği olmayan ses kartı sürücülerine ASIO özelliği kazandıran eklenti.

▪ \DelphiASIOVST

Delphi için ASIO bileşeni.

▪ \PNGImage

Delphi için PNG bileşenleri.

▪ \Scl

Scala arşivinden alınmış dizi dosyaları.

▪ \Tablet131

Delphi için tablet (WinTab) bileşeni.

❖ \Metin

PDF formatında proje metni.

❖ \Plaka

Plaka yazılımı ve kaynak kodu.