

EREN
KARACAN

FMY-WEB.GITHUB.IO

Ses Programlamaya Giriş

Pure Data 0.1

İçindekiler

Genel Bilgi	3
Pure Data Temelleri	4
Pure Data Nedir?.....	4
Pure Data'ya Giriş	4
Kontrol Kutuları	6
Bağlantılar, Dört İşlem, Karşılaştırmalar ve Metronom	7
Ses Sentezleme Araçları	9
Ses Araçları ve Bilgi Türleri.....	9
Line Objesi.....	9
Vline~ Objesi	10
Sinyal Görselleştirme	11
Panorama	11
Çalışma Soruları	12
Eklemeli (Additive) Ses Sentezleme	13
Genlik Modülasyonu (Amplitude Modulation)	15
Frekans Modülasyonu (Frequency Modulation).....	16

Genel Bilgi

Ses Programlamaya Giriş dersinde ses programlamaya ilişkin temel kavramlar üzerinde durulacaktır. İşlenecek konuların daha basit bir şekilde anlaşılması ve teknik detaylardan çok altta yatan mantığın çözümlenebilmesi için açık kaynaklı bir görsel programlama dili olan Pure Data kullanılacaktır. Kullanım kolaylığı için Pure Data'nın çeşitli özellikler eklenmiş bir versiyonu olan "Purr Data" yazılımı ile çalışılacaktır. Bu yazılım <https://github.com/jonwwilkes/purr-data/releases> adresinden indirilebilir.

Latest release

2.14.1

jonwwilkes released this 11 days ago · 88 commits to master since this release

- fixed case sensitive searches in the search browser
- fixed broken keyboard navigation in the search browser
- added bookmarking tool for search browser to customize the home page
- various other search browser fixes
- adjusted scale on the vu meter to match position of vanilla pd
- removed excess y size in the VU dialog
- cached the search browser index so that it loads much more quickly on subsequent uses
- update translations
- fixed GUI presets so they work with the new properties dialogs
- added possibility for user-defined GUI preset css files
- changed hotkey for temporary run mode. (Added a console message about it when the user clicks the old hotkey for temporary run mode.)
- fixed a regression in 2.14.0 with the find tab at the bottom of the canvas window
- updated build instructions to reflect changes in MSYS2 dependencies
- visual improvements to the preferences window
- dropped ancient nw.js version 0.14.7 on Windows for a more recent version. This means the GUI will no longer work with Windows XP.
- added possibility to build on Windows as a 64-bit binary
- added tiled background for patches when in editmode (may be turned off in the preferences)

Assets

osx_10.11-x86_64-dmg.zip	109 MB
osx_10.8-x86_64-dmg.zip	109 MB
windows-i386.zip	100 MB
Source code (zip)	
Source code (tar.gz)	

İlerleyen derlerde Pure Data içerisindeki objeler ve mesajlar, bu obje ve mesajların sentezleme ve analiz uygulamalarındaki kullanım şekilleri ve temel dijital sinyal işleme kavramları üzerinde durulacaktır.

Pure Data Temelleri

Pure Data Nedir?

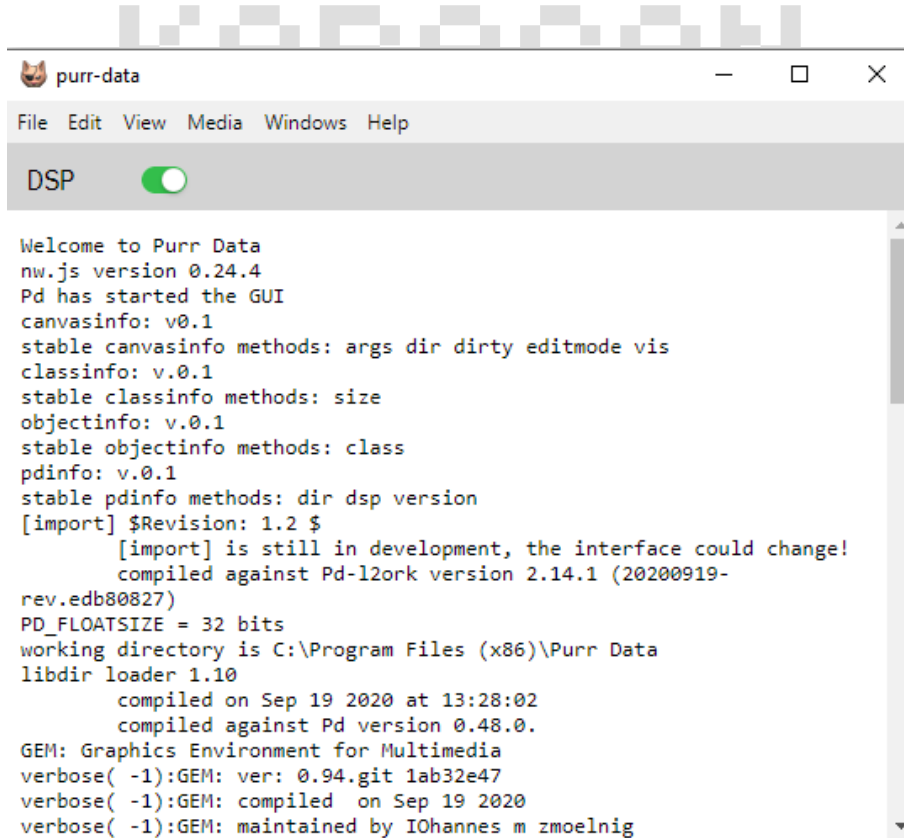
Pure Data (kısaca PD), ilk versiyonu Miller S. Puckette tarafından 1996 yılında etkileşimli bilgisayar müziği ve multimedya gibi alanlarda kullanılması amacıyla geliştirilmiş bir görsel programlama dilidir. Yazılımın asıl geliştiricisi Puckette olsa da, açık kaynaklı bir proje olan PD zaman içerisinde birçok farklı geliştirici-kullanıcının katkısıyla günümüzdeki haline evrilmiştir.

Ses programlamaya görsel bir yaklaşım sunan PD, hem programlamanın (ve özellikle ses programlamanın) daha kolay anlaşılmasında, hem de ses (genellikle sentez) ve görüntü uygulamalarının tasarımının kolaylıkla gerçekleştirilmesinde sağladığı rahatlık dolayısıyla geniş çevrelerce tercih edilmektedir.

Pure Data'ya Giriş

PD'yi açtığınızda karşınıza “ana ekran” çıkacaktır. Bu ana ekran çeşitli ayarlama ve yardım sekmeleri ile birlikte bir adet de konsol sunmaktadır. Bu konsol açılışta PD'nin çalıştırılma sürecine dair çeşitli bilgileri göstermektedir. Ayrıca tasarlanan “patch”lerdeki (Pure Data'da tasarlanan ufak programlara patch denmektedir) yazı ile iletişim kurması gereken öğelerin yazılı çıktılarının okunabildiği ekran da bu konsoldur.

Sol üstteki “File” sekmesinden “New” seçeneği seçildiğinde yeni bir patcher/canvas penceresi açılacaktır. Bu pencere tasarlanacak patch'in içinde bulunacağı ana ortamdır.



PD’de sinyal işlemlemeyi sağlayan araçlar özel olarak tasarlanmış kutulardır. Bu kutular dört ana başlıkta incelenebilir:

1. Mesaj kutuları
2. Number/Atom kutuları
3. Obje kutuları
4. Comment kutuları

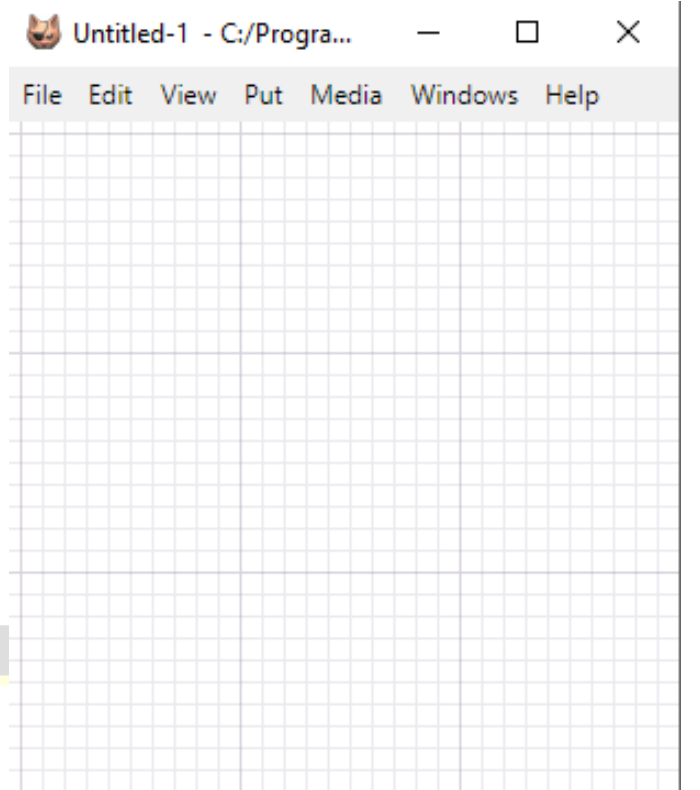
PD’deki araçların her biri özel bağlantı noktalarına sahiptir. Genellikle üst taraftaki noktalar “input”, yani veri girişi; alt taraftaki noktalar ise “output” yani veri çıkışı için kullanılmaktadır. Bu bağlantı noktalarından kabul edilen ve gönderilen mesajların türü aracın karakterine özgüdür. Bağlantılar, bağlantı noktalarına tıklanıp sürükleyip bırakılarak yapılmaktadır.

Mesaj kutuları sayısal veya yazılı bilgi içeren kutulardır. Sol alttaki output noktasından taşıdıkları bilgiyi gönderirler. Bu bilgiyi göndermeleri tipik olarak iki aksiyona bağlıdır: bunlardan biri mouse ile mesaj kutusunun üzerine tıklanması, diğeri ise sol üstteki input noktasından bir “bang” mesajının alınmasıdır (bang mesajlarına aşağıda değinilecektir).

Number/atom kutuları ise sayısal verilerin taşınması ve kolaylıkla değiştirilebilmesi için kullanılan kutulardır. Atom kutularındaki sayısal veriler mouse ile tıklayıp sürükleyerek değiştirilebilir. Veri değişimi meydana geldiğinde yeni değer kutunun altındaki output noktasından gönderilecektir.


Not: PD’de “edit mode” ve “play (çalma) mode” olmak üzere iki farklı mod vardır. Edit modu çeşitli öğelerin yerleştirilip bağlanması için kullanılırken çalma modu bu objeler ile etkileşime geçmek için kullanılır. Bu modlar arasındaki geçiş Ctrl+E (Mac kullanıcıları için Command+E) kısayoluyla gerçekleştirilir.

Objeler ise pek çok farklı fonksiyona sahip araçlardan meydana gelmektedir. Bu araçların kullanılabilmesi için öncelikle bir obje kutusunun oluşturulması (ctrl/cmd+1) ve kutunun istenen objenin ismi ile isimlendirilmesi gerekmektedir. PD’de birçok farklı işleve sahip birçok obje vardır ve her objenin input/output noktaları, objenin özel çalışma prensibine göre şekillenmektedir. Bu sebeple bir objenin kullanılabilmesi için objenin davranışının ve giriş/çıkışta kullandığı verilerin doğru olarak bilinmesi gerekmektedir. Konsola yazı yazmak için kullanılan obje “print” objesidir. Print objesi tek bir input noktasından veri alır ve kendisine ulaşan veriyi konsola yazar.



Hello World
mesaj kutusu

32
number/atom kutusu

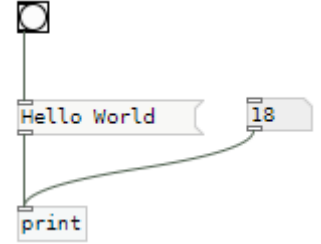

tanımlanmamış boş obje

print
print objesi

Not: “Bang” mesajları tetikleyici mesajlar olarak düşünülebilir. PD’deki birçok araç belirli işlemleri gerçekleştirmek için bang mesajlarını bekler. Bang mesajları bang objeleri vasıtasıyla gönderilebilir. Bunun için Ctrl+1 kısayolu ile bir obje oluşturulmalı ve “bng” olarak isimlendirilmelidir. İlerleyen haftalarda bu konu daha detaylı işlenecektir.

Comment kutuları tamamen kullanım kolaylığına yönelik araçlardır. Kullanıcının belirli noktalara düşmek isteyebileceği notlar, bu kutular vasıtasıyla düşülebilir. Yukarıdaki görsellerdeki “mesaj kutusu”, “print objesi” gibi açıklamalar comment kutuları ile yazılmıştır. Bu kutulara yazılanlar tamamen kullanıcıya not düşmek amacıyla. Herhangi bir özel işlevleri yoktur.

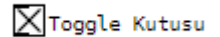
Sağdaki görselde bir bang objesi, bir mesaj kutusu, bir sayı/atom kutusu ve bir print objesinden meydana getirilmiş bir patch sunulmuştur. PD’de yazılabilecek belki de en basit patch’lerden biri olan bu patch ile (çalma modundayken) bang mesajına veya mesaj kutusuna tıklandığında print objesine “Hello World” yazısı gönderilecektir. Yazı print objesinin üst tarafındaki input noktasına ulaştığında ise konsola yazılacaktır. Ayrıca (yine çalma modundayken) atom kutusuna tıklanıp yukarı aşağı sürüklendiğinde kutudaki değer değişecek, değişen her yeni değer yine print objesine gönderilecek ve konsola yazılacaktır.



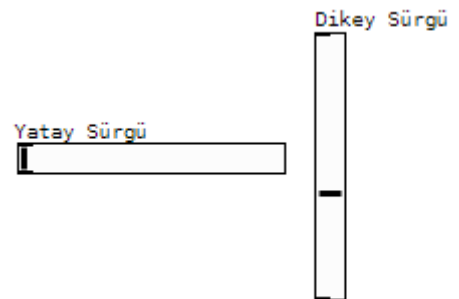
Kontrol Kutuları

Pure Data’da çeşitli farklı kontrol türleri ve özel veri aktarım yöntemleri için tasarlanmış belirli kutular mevcuttur. Tasarlanacak algoritmalarda bu kutuların kullanımı pek çok açıdan kolaylık sağlayacaktır.

“Toggle” kutusu açık veya kapalı olma durumuna bağlı olarak output noktasından 1 ve 0 değerleri gönderir. Açık olma durumunda göndereceği 0 olmayan değer ise sağ tıklanarak açılabilen “Properties” (Özellikler) sekmesinden değiştirilebilmektedir.

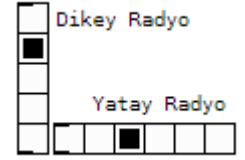


“Slider” kutusu ise belirli sayısal değerlerin bir sürgü vasıtasıyla üretilmesi içindir. Vslider (Vertical Slider – Dikey Sürgü) ve Hslider (Horizontal Slider – Yatay Sürgü) olarak iki farklı tür mevcuttur. Özellikler sekmesinde birkaç seçenek mevcuttur. “minimum-maximum” alanları sürgünün alacağı en düşük ve en yüksek değerlerin girildiği alanlardır. “init” seçeneği sürgünün .pd uzantılı dosya her açıldığında belirli bir değerden başlaması için kullanılır.



“logarithmic scaling” seçeneği ise sürgünün ivmesinin lineer mi logaritmik mi olacağını kontrol eder. “steady on click” seçeneği ile ise sürgünün değerinin sürgü üzerinde bir noktaya tıklanarak değiştirilip değiştirilemeyeceğini ayarlar.

“Radio” kutusu ise birçok farklı sayısal değer için farklı anahtarlar barındıran bir kontrol kutusudur. Anahtarların sayısal değerleri 0’dan başlar ve birer birer artar. Vradio ve Hradio türleri mevcuttur. Radio’nun barındıracağı anahtar sayısı özellikler sekmesinden ayarlanabilir.



Bağlantılar, Dört İşlem, Karşılaştırmalar ve Metronom

Pure Data’daki kutuların farklı soketleri farklı işlevlerde kullanılır. Bu işlevler genellikle kullanılan objeye özgü olsa da, genel bir kuraldan bahsetmek mümkündür. Kutuların en solunda bulunan giriş soketi, “sıcak giriş/soket” olarak, diğer giriş soketleri ise “soğuk girişler/soketler” olarak adlandırılır. Sıcak soketlere gelen bilgiler, objelerin belirli işlemleri yapmalarını sağlar. Soğuk girişlere gelen bilgiler ise bu işlemler yapılırken kullanılacak parametreler olarak objelerde depolanır.

Pure Data’da yapılan bağlantıların sırası da sinyal önceliği bakımından oldukça önemlidir. Genel bir kural olarak Pure Data sinyal önceliğini hesaplarken kullanıcının yaptığı bağlantı sırasını gözetir.

Sıcak/soğuk soketleri ve sinyal önceliğini anlamak için dört işlem objeleri ile çalışılabilir. Pure Data’da basit dört işlem basit objelerle yapılır.



Görüldüğü üzere dört işlem objelerinde birer adet sıcak ve soğuk giriş bulunur. Bu girişlerdeki bilgilerin kullanım şekli şu şekilde hayal edilebilir:

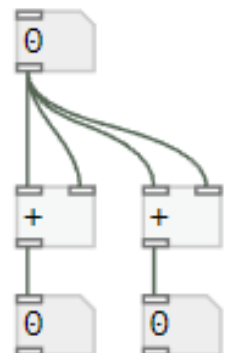
soğuk giriş - toplanacak sayıyı hafızaya depola

sıcak giriş - gelen sayı ile hafızadaki sayıyı topla ve sonucu çıkış soketinden yolla







Doğal olarak böyle bir tasarımda hangi bilginin daha önce geldiği büyük önem kazanmaktadır. Sıcak girişten bir sayı göndermeden önce toplanacak sayının soğuk girişten gönderilmesi gerekmektedir. Bu öncelik ise bağlantı önceliği ile sağlanabilir. Pure Data’da hangi bağlantının önce yapıldığını görsel olarak ayırt etmek mümkün değildir. Dolayısıyla tıpatıp aynı görünen iki bağlantıdan biri doğru, diğeri yanlış olabilmektedir. Bu sebeple (özellikle sıcak-soğuk bağlantılar yapılırken) kablolama önceliğine her zaman dikkat edilmeli, tam olarak anlaşılamayan bir sorunla karşılaşıldığında ise kablolama sırasına şüpheyle yaklaşılmalıdır.

Örnek Çalışma

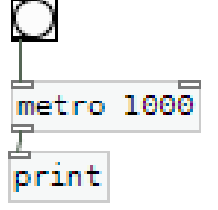
Yandaki toplama işlemlerinden biri doğru diğeri yanlış sonuç vermektedir. Bunun sebebi ne olabilir? Yanlış sonuç veren sistem nasıl çalışmaktadır?



Karşılaştırma objeleri programlamada yaygın olarak kullanılan karşılaştırma opratörlerinden oluşmaktadır ve dört işlem objeleri ile aynı mantıkla çalışmaktadır.

 eşittir	 büyüktür	 büyük eşittir
 eşit değildir	 küçüktür	 küçük eşittir

Ses programlamada en çok uğraşılacak konulardan biri de zamandır. Daha açık bir ifade ile belirli sürelerde tekrarlayan periyodik hareketlerdir. Neyse ki Pure Data periyodik işlemlerin yapılmasını çok kolaylaştıran bir “metro” objesi bulundurmaktadır. Metro objesi, oluşturulması esnasında yazılan sayıyı veya daha sonraki bir zamanda soğuk girişten gönderilen sayıyı argüman olarak alır, ve milisaniye bazındaki aralıklarla bang mesajı üretir.



Örneğin “metro 1000” objesi saniyede bir, “metro 5000” objesi beş saniyede bir bang mesajı gönderecektir. Metro objesinin çalıştırılması için öncelikle sıcak girişten bir toggle ya da bang mesajı alması gereklidir.



EREN
KARACAN

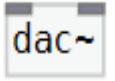
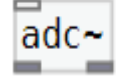
PMY-WEB.GITHUB.IO

Ses Sentezleme Araçları

Ses Araçları ve Bilgi Türleri

Pure Data, çeşitli farklı ses sentezleme uygulamaları göz önünde bulundurularak meydana getirilmiş bir programlama dilidir. Dolayısıyla ses sentezlemeye dair pek çok farklı araçla karşılaşılabilir.

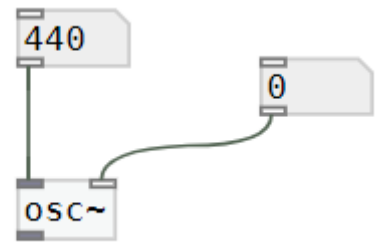
Sentezleme araçlarından önce bahsedilmesi gereken araçlar, herhangi bir ses sinyalinin Pure Data içerisinde işlenebilmesini, sonra da dinlenebilmesini sağlayan dönüştürücü (converter) araçlarıdır. Bu dönüştürücüler müzik teknolojisine dair pek çok alanda görmeye alışık olduğumuz analog-dijital dönüştürücü ve dijital-analog dönüştürücüdür. Pure Data'da bu dönüştürücüler “adc~” ve “dac~” objeleri olarak kullanılırlar.



Bu objelerle ilgili göze çarpan iki özellikten bahsedilebilir. Öncelikle iki objenin de isminin yanında bir tilde (~) işareti bulunmaktadır. Öte yandan giriş ve çıkış soketlerinden bazıları daha koyu renklidir. Bu iki durumun sebebinden bahsetmek için öncelikle Pure Data'da iletilen bilgi tiplerine bakılması gerekmektedir.

Pure Data'da iki farklı bilgi tipi ile çalışılmaktadır. Bunlardan bir tanesi çeşitli araçların nasıl ve ne zaman çalışacağını belirleyen *kontrol bilgisi*, diğeri ise belirli bir sample rate'e bağlı olarak iletilen ve ses sample'larından oluşan *ses sinyali*dir. Şimdiye kadar yalnızca *kontrol seviyesinde* işlemler ve bağlantılar üzerinde çalışıldı. Bu sebeple kullanılan objelerde tilde işareti, koyu renkli soketler ve kalın bağlantı kabloları ile karşılaşmadı. Ancak bu noktadan sonra sıklıkla çalışacağımız *sinyal seviyesinde* bütün bunlarla sıkça karşılaşacağız. Zira çeşitli objelerin isminin yanında bulunan tilde işareti bu objelerin ses sinyali ile çalıştığını, koyu renkli soketler ve kalın bağlantı kabloları ise taşınan bilginin ses sinyali olduğunu göstermektedir.

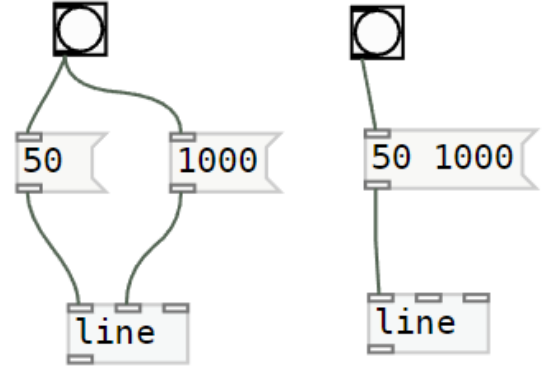
Pure Data'da sinüs (ve kosinüs) dalgalarının üretilmesini sağlayan obje “osc~” objesidir. Bu objenin iki giriş soketi bulunmaktadır. Soldaki sıcak giriş soketi sinüs dalgasının istenen frekansının, sağdaki soğuk giriş ise istenen faz miktarının ayarlandığı kısımlardır.



Line Objesi

Ses sentezleme ve programlama uygulamalarının birçoğunda kullanılan temel araçlardan biri, değerlerin belirli sürelerde belirli ölçüde artması ya da azalmasıdır. Bu durumun en basit örneği olarak fade-in ve fade-out uygulamaları gösterilebilir. Bu uygulamalarda ses seviyesi bir başlangıç değeri ve bir de sonuç değeri arasında belirli sürede bir değişime uğramaktadır. Pek çok farklı durumda kullanılabilen ve pek çok farklı duyuma imkân sağlayabilen bu araçlar aslında oldukça basit araçlardır. Hatta bu araçların en basit ifadeyle “bir sayısal değer belirli sürede başka bir sayısal değere dönüşmesi” olayından ibaret olduğu söylenebilir.

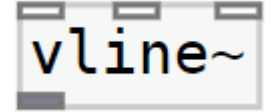
Bu işlemi gerçekleştiren Pure Data objesinin ismi “line”dır. Line nesnesi bir sayısal değeri verilen süre içerisinde doğrusal bir ivmeyle başka bir sayısal değere çevirir. Nesnenin sıcak soketi varılacak sayı değerini belirler, ikinci soket ise bu sayısal değere kaç milisaniyede varılacağı bilgisini alır. Üçüncü soket grain size soketidir ve çıkış bilgisinin kaç milisaniyede bir gönderileceğini belirler. Örneğin 0’dan 50’ye 2000 ms’de çıkacak bir rampa ayarlarsak ve bu rampanın grain size’ını 1000 ms olarak belirlersek sonuç kutusu 1 saniye boyunca 0’da bekleyecektir, 1. saniyede 25’e eşit olacaktır ve 2. saniyede 50 değerine ulaşacaktır. Default grain size 20 ms’dir. Bir line objesi birden fazla farklı şekilde yazılabilir. Yandaki iki yazım birbirinin aynı sonuçlar vermektedir.



Sinyal türleri farklı olduğu için yanda gördüğümüz “line” objesinin ses sinyalleriyle çalışması mümkün değildir. Ses sinyaliyle çalışmak için “line~” objesi kullanılmalıdır.

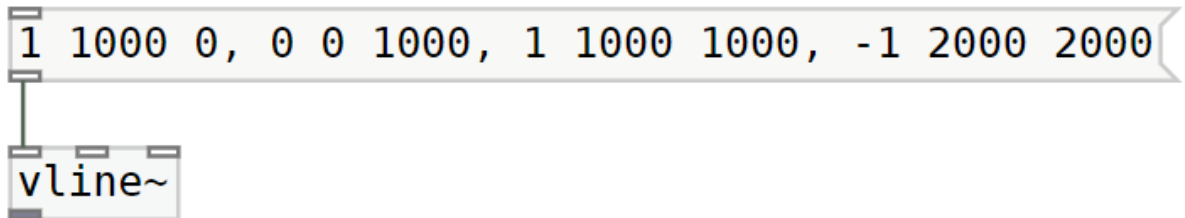
Vline~ Objesi

Ses sinyallerinin zaman içerisinde değişimden bahsettiğimiz zaman akla gelen ilk kavramlardan biri de genlik zarfıdır. Line~ objesi ile basit bir ADSR zarfı yaratmak mümkün olsa da line objesi bu iş için çok elverişli değildir. Bu sebeple Pure Data içerisinde başka bir obje tanımlanmıştır. Bu obje vline~ objesidir.



Vline objesine gönderilen mesajlar (en fazla) üç parçadan oluşur. Birinci parça hedef değeri, ikinci parça hedefe ulaşılacak süreyi (belirtilmezse sıfır olarak kabul edilir), üçüncü parça ise hedefe hareketin başlamasından önce beklenmesi gereken süreyi ifade eder (belirtilmezse sıfır olarak kabul edilir).

Bu durumda bir değer 1000 ms içerisinde 1’e çıkması, sonra aniden 0’a düşmesi, tekrar 1000 ms içerisinde 1’e çıkması ve son olarak 2000 ms içerisinde -1’e düşmesi istendiğinde şu şekilde bir mesaj gönderilebilecektir:



Görüldüğü üzere yukarıdaki dört aşamalı işlem dört öge barındıran tek bir mesaj ile gerçekleştirilebilmektedir. Toplu mesajın her ögesi sırayla çalıştırılacak komutları içermektedir. Dikkat edilmesi gereken, mesajların üçüncü parçasını oluşturan bekleme süresidir. Tek mesajdaki tüm komutlar aynı zaman çerçevesini paylaşır. Bu

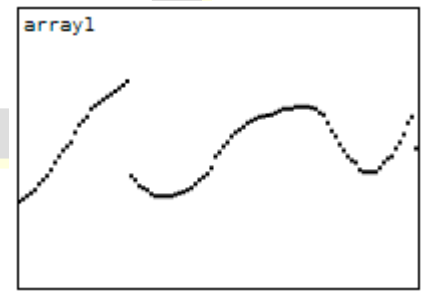
sebeple eğer ilk hareket 1000 ms sürecekse ikinci hareketin bekleme süresi 1000 ms olmalıdır. Aynı mesajdaki ikinci hareket 1500 ms sürecekse de, üçüncü mesajın bekleme süresi 2500 ms olmalıdır.

Sinyal Görselleştirme

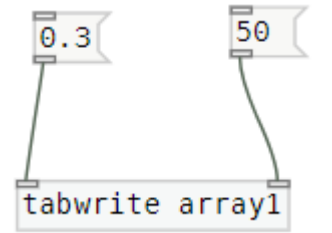
Ses sentezleme pratiği büyük oranda kulağa ve duyuma dayansa da, özellikle temel işlemler söz konusu olduğunda çeşitli teorik bilgilere dayalı uygulamalar devreye girebilmektedir. Bu teorik bilgilere dayalı uygulamaların doğru bir şekilde uygulanıp uygulanmadığını anlamakta sıklıkla kullanılan bir yöntem de sinyal grafiğini incelemektir. Dolayısıyla eldeki ses sinyalinin grafiğe dökülüp görselleştirilmesi oldukça kullanışlıdır.

Pure Data’da basit görselleştirme için iki araçtan yararlanılır. Bunlardan birincisi “tabwrite” objesi, ikincisi ise “array”dir.

Array’ler aslen aynı türden değerler barındıran kümelerdir. Pure Data’da da bu durum aynıdır ancak array’lerin kullanımları biraz daha farklıdır. Pure Data’da arrayler genellikle bu aynı türden değerler barındıran (örneğin -1’den 1’e kadar float değerlerinden meydana gelen ses sinyalleri) kümelerin görselleştirilmesi için kullanılır. Başka bir ifadeyle bu kümelerin grafiğini çıkarmaya yarar.



Tabwrite objeleri ise belirli değerlerin array’lere yazılması için kullanılır. Soldaki sıcak soket dikey eksendeki konumu belirleyen değeri kabul eder. Sağdaki soğuk sokete gönderilen bilgi ise değerin yatay eksendeki pozisyonunu belirleyen “indeks”i oluşturur.

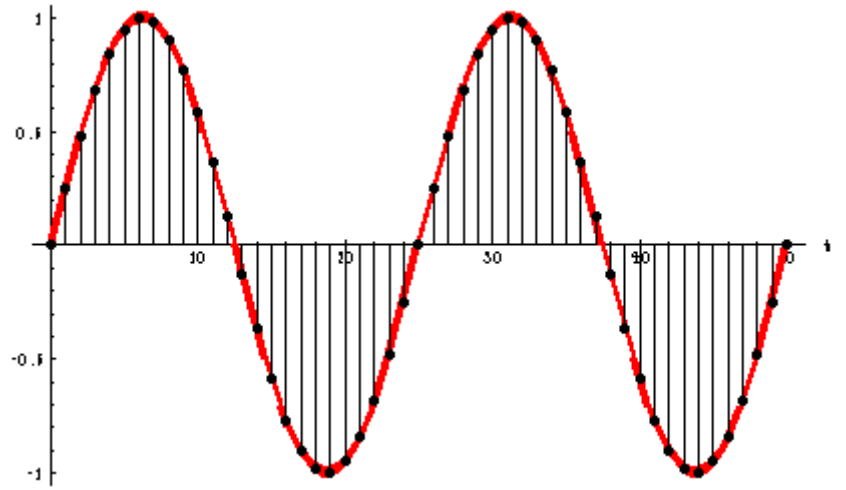


Panorama

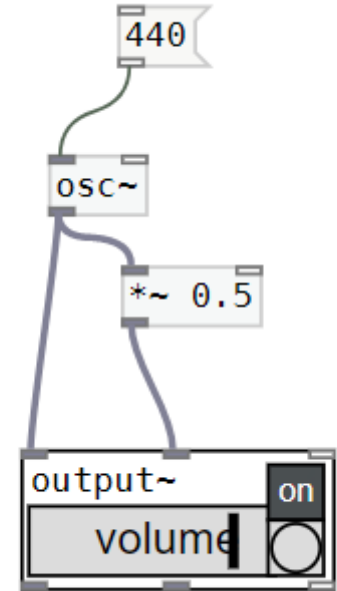
Ses teknolojisinde kullanılan en temel araçlardan biri de panlamadır. Panlama işlemi ses sinyalinin bir stereo ya da çok kanallı ses düzlemine dağıtılarak bir ses panoramasının oluşturulmasından ibarettir. En yaygın kullanılan çok kanallı ses düzlemi stereofonik düzlem olduğu için bu derste de stereo panlama üzerine çalışılacaktır.

Panlama işlemi en basit haliyle “ses sinyalinin sağ ve sol kanallara farklı kuvvetlerde gönderilmesi” olarak tanımlanabilir. Bu tanıma göre bir panlama yapmak istediğimizde “output~” ya da “dac~” nesnelerine gönderdiğimiz ses sinyallerinin seviyeleri üzerinde bir manipülasyonda bulunmamız gerekecektir.

Dijital ses sinyalleri üzerinde manipölasyonda bulunmadan önce ses sinyallerinin dijital düzlemde nasıl ifade edildiğini hatırlayalım. Dijital ses analog ses sinyalinin belirli sayıda örnekleme noktasından ve bu örneklerin -1 ve +1 arasındaki sayılar olarak ifade edilmesinden ibarettir. Dolayısıyla dijital ses sinyalinin bir dizi sayıdan ibaret olduğunu söylemek de mümkündür. Bu durumda sayılar üzerinde yapabildiğimiz her işlem dijital ses sinyalleri üzerinde de yapılabilecektir.



Böylece bir stereo panlama işleminin de ne şekilde yapılacağı oldukça belirginleşmektedir. Tekrar ifade etmek gerekirse, panlama işlemi en temel haliyle çıkış kanallarına giden seslerin seviyelerinin, başka bir ifadeyle genliklerinin değiştirilmesi demektir. Bu durumda herhangi bir çıkış kanalına giden sesi oluşturan sample'ların ikiye bölünmesi (yukarıdaki görselden hareket etmek gerekirse genliği 1 ve -1 değil 0,5 ve -0,5 arasındaki mesafeden oluşan yeni bir sample oluşturulması) yeterli olacaktır.



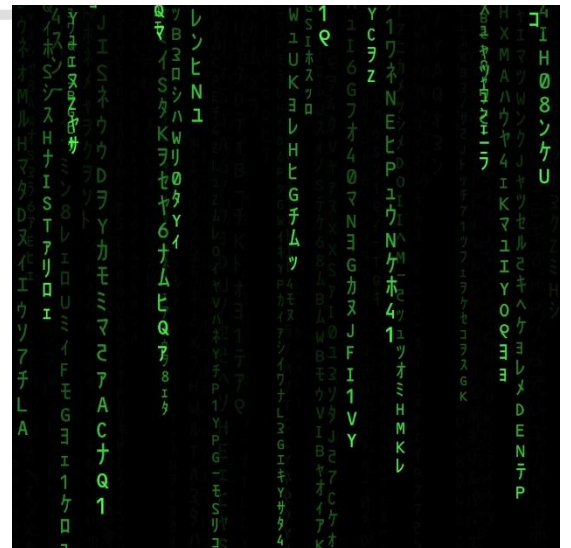
Çalışma Soruları

Soru 1

“tabwrite” objesini kullanarak bir grafik tasarlayın. Grafik “Matrix” filminde popülerleşmiş “dijital yağmur”u, başka bir ifadeyle kayan yazıları çağrıştırmalıdır.

Soru 2

İki adet “osc~” objesi kullanın. Bu objelerin ortaya çıkardığı ses sinyallerinin fazlarını birbirlerini sıfırlayacak şekilde ayarlayın.



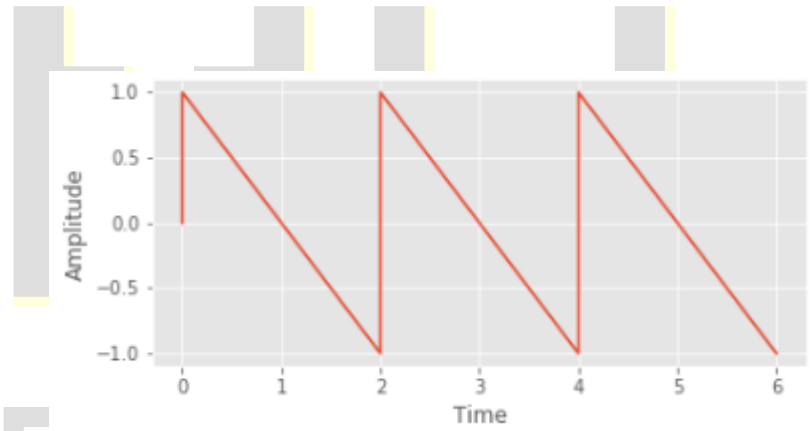
Eklemeli (Additive) Ses Sentezleme

Eklemeli sentezleme, en temel ve eski ses sentezleme yöntemlerinden bir tanesidir. Eklemeli sentezlemede tüm karmaşık dalgaların basit sinüs dalgalarının toplamı olarak ifade edilebileceğini söyleyen Fourier teoremine dayanılır. Çeşitli tınılar elde etmek için farklı frekans ve genlik değerlerine sahip sinüs dalgaları LFO ve ADSR envelope gibi çeşitli modülatörlerce modüle edilerek toplanır. Elde edilen ses birçok sinüs dalgasının birbirine eklenmesi ile meydana getirildiği için “eklemeli sentezleme” ismi kullanılmıştır.

Şimdiye kadar edindiğimiz Pure Data bilgisiyle çeşitli basit dalga formlarını eklemeli sentezleme yöntemiyle sentezlemek zor olmayacaktır. Sentezlemede kullanılan temel dalga formları **testere**, **kare** ve **üçgen** dalgalarıdır.

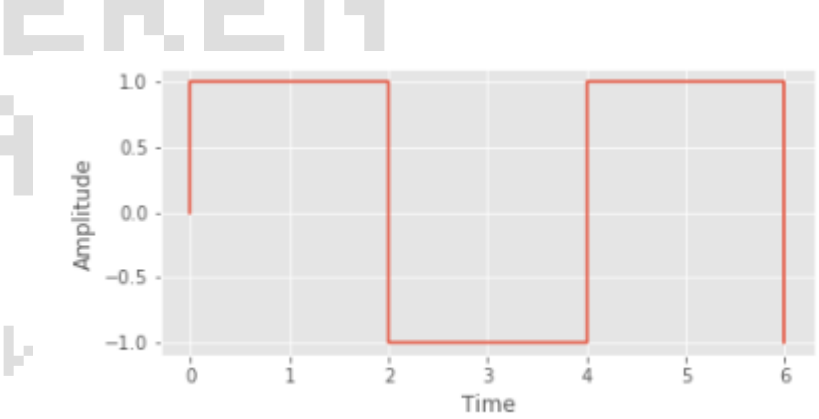
Testere Dalga

Harmonikler: Tüm harmonikler
Genlikler: $1 / \text{Harmonik numarası}$



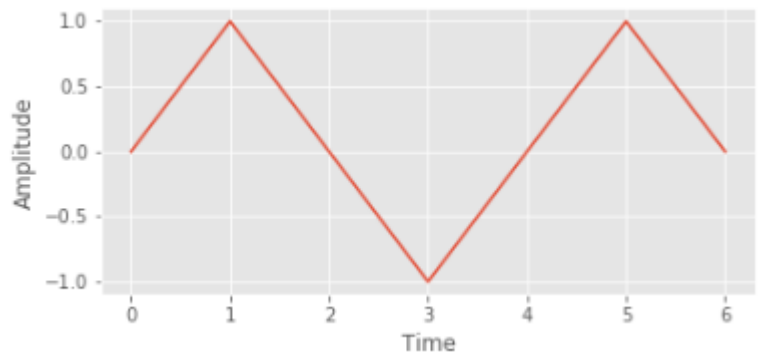
Kare Dalga

Harmonikler: Tek sayılı harmonikler
Genlikler: $1 / \text{Harmonik numarası}$

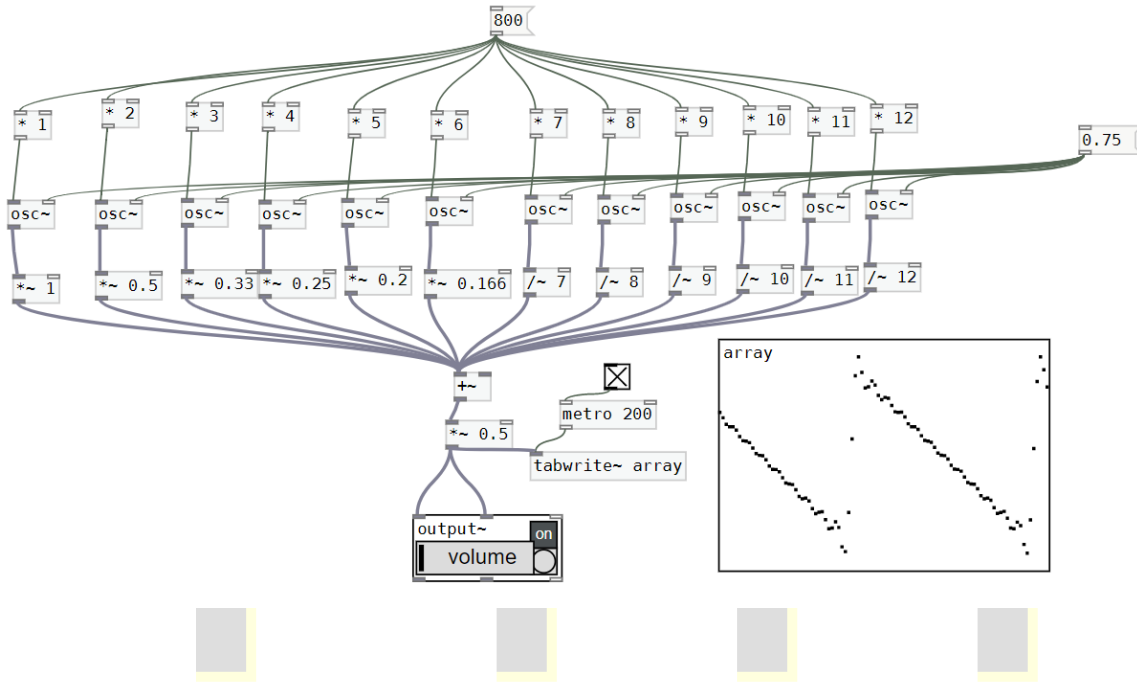


Üçgen Dalga

Harmonikler: Tek sayılı harmonikler
Genlikler: $1 / (\text{Harmonik numarası})^2$
Faz: Her iki harmonikten bir tanesi 180 derece ters fazlı



Unutulmamalıdır ki verilen formüller ile dalga formlarının eklemeli yöntemle kusursuz olarak sentezlenmesi için teorik olarak sonsuz sayıda doğuşkan sentezlenmelidir. Ancak bu mümkün olmadığı için teker teker artırılan doğuşkanlarla sinüs dalgası formundan karakteristik dalga formuna geçiş gözlenebilir.



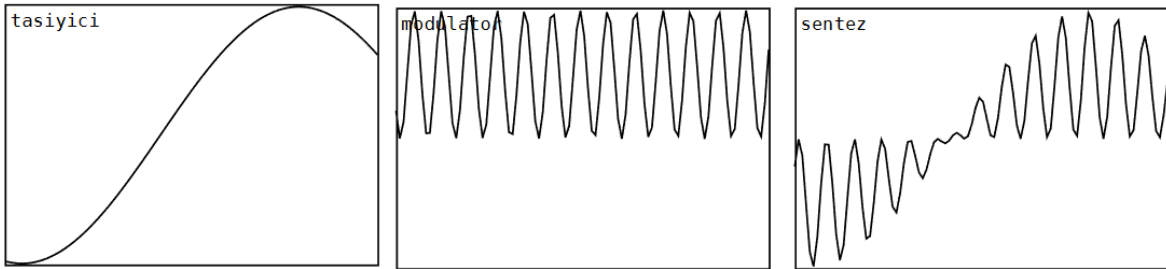
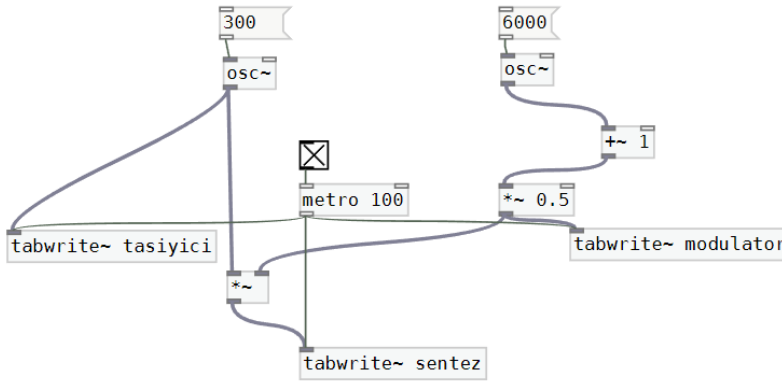
EREN
KARACAN

PMY-WEB.GITHUB.IO

Genlik Modülasyonu (Amplitude Modulation)

Genlik modülasyonu bir sinyalin genliğinin başka bir sinyalin genliği ile kontrol edilmesi (ya da modüle edilmesi) prensibine dayanan bir sentez yöntemidir. Modüle edilen sinyale “carrier” (taşıyıcı), modüle eden sinyale de “modulator” (modüle edici/modülatör) sinyal ismi verilir. Klasik genlik modülasyonu sentezinde carrier sinyal de modulator sinyal de osilatörlerden elde edilir. Ancak modülatör sinyalin enstrüman ya da vokal gibi çeşitli kaynaklardan elde edilmesi de mümkündür. Oldukça düşük frekanslı modülatörlerle yapılan genlik modülasyonuna “tremolo”, bipolar sinyallerle yapılan genlik modülasyonuna “ring modülasyonu” ismi verilmiştir.

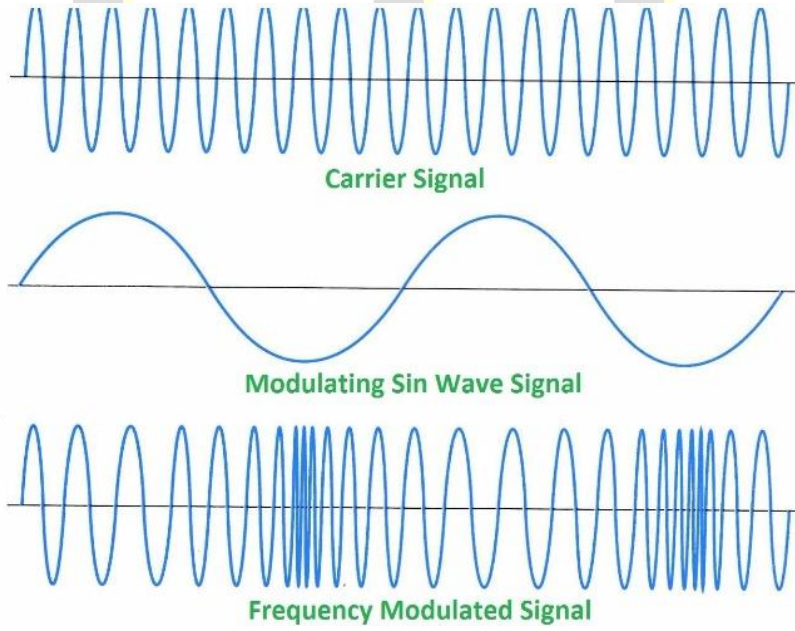
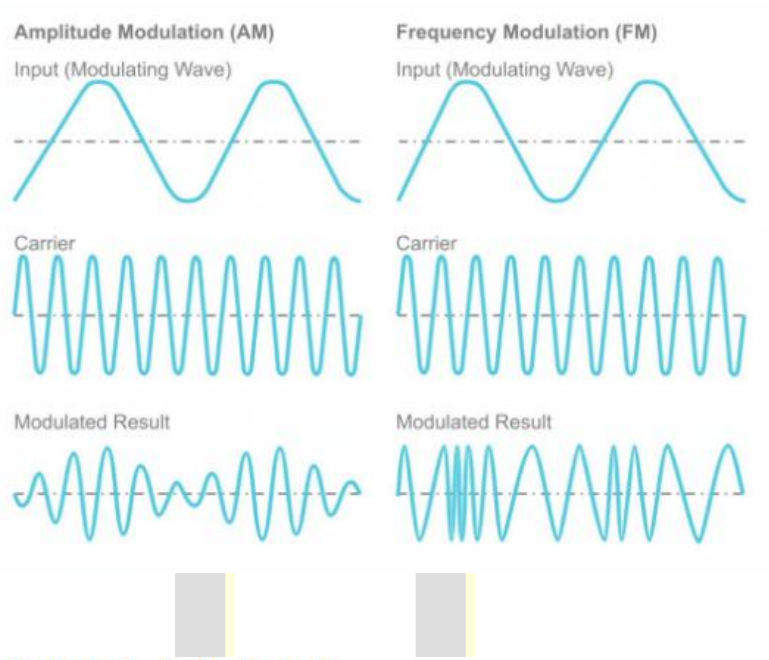
Klasik genlik modülasyonunda bir osilatörün genliği diğer osilatörün genliği ile modüle edilir. Bu senaryoda taşıyıcı sinyalin genliği 0 ve 1 arasındaki değerlerle modüle edilmektedir. Zira bir dijital ses sinyalinin genliği her zaman 0 ve 1 arasındaki değerlerden meydana gelmektedir. Bu sebeple modülatör olarak kullanılan sinyal eğer bir ses sinyaliyse, öncelikle sinyal değerlerine 1 eklemek ardından tüm sinyal değerlerini 0 ve 1 arasında tutmak için 0.5 ile çarpmak gereklidir. Örnek bir genlik modülasyonu aşağıdaki şekilde sunulmuştur.



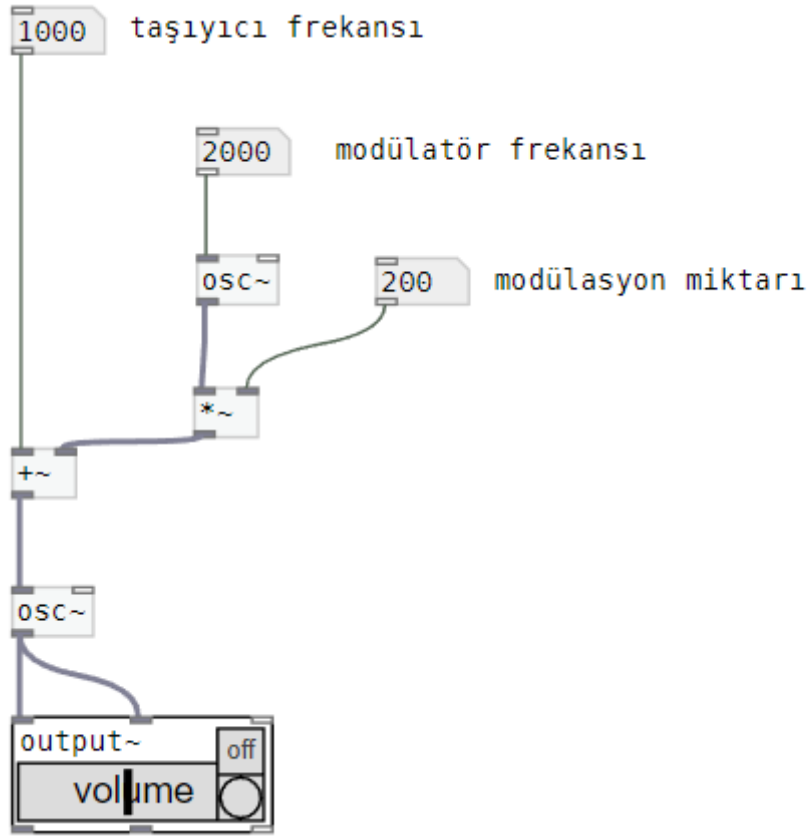
Genlik modülasyonunda iki adet “sideband” (yan bant) frekansı oluşur. Bu frekanslar $f_2 + f_1$ ve $f_2 - f_1$ frekanslarıdır. Yani yukarıdaki örnekte 6300 Hz ve 5700 Hz olmak üzere iki yeni frekans meydana gelmiştir.

Frekans Modülasyonu (Frequency Modulation)

Ses sentezleme teknikleri arasında en yaygın kullanılanlardan biri, frekans modülasyonu sentezidir. Frekans modülasyonu sentezinde, genlik modülasyonu sentezine benzeyecek şekilde, bir taşıyıcı (carrier) ve bir de modülatör (modulator) sinyal bulunmaktadır. Ancak genlik modülasyonundan farklı olarak bu sefer modülatör sinyalin genliği taşıyıcı sinyalin genliğini değil, frekansını modüle etmektedir.



Şekilde basit bir FM sentez örneği sunulmuştur:



Tahmin edileceği üzere, modüle etmek istediğimiz değer frekans olduğu için, amacımız osilatör objesinin sıcak soketine girecek olan sinyalin dinamik olarak değişmesidir. Bu değişim ise belirlenen bir frekans değerinin (örnekte 1kHz) başka bir osilatörden elde edilen sinyale bağlı olarak artıp azalmasıyla elde edilecektir (başka bir ifadeyle osilatöre gidecek 1000 değerinin 2000 Hz hızında belli bir oranda değişmesi istenmektedir). Ancak modülatör sinyalin taşıyıcı sinyalin frekansını ne miktarda değiştireceği (modülasyon miktarı) de gözden kaçırılmamalıdır.

Bir an için yukarıdaki şekildeki “200” değerine sahip modülasyon miktarının “1” değerine sahip olduğunu düşünelim. Bu durumda modülatör sinyalinin taşıyıcı sinyalin frekansına yapacağı etki, taşıyıcı sinyalin frekansının saniyede 2000 kere -1 ve +1 değerleri ile toplanması olacaktır. Yani taşıyıcı sinyalin frekansı 1001 ve 999 Hz arasında saniyede 2000 kere salınarak gidip gelecektir.

Tahmin edeceğiniz üzere, bu kadar küçük bir değişimin ortaya çıkaracağı sonuç pek duyulabilir olmayacaktır. Ancak bu salınımın 1200 Hz ve 800 Hz arasında gerçekleştiği bir durumda elde edilen sonuçları duymak oldukça rahat olacaktır. Bu sebeple modülatör sinyalin taşıyıcı sinyalin frekansına göstereceği etkiyi artırmak için sinyalin taşıdığı değerler (başka bir düşünceyle sinyalin genliği) çarpılarak artırılmıştır. Böylece taşıyıcı sinyalin frekansındaki değişim çok daha fark edilebilir olacaktır.

FM sentezinde ortaya çıkan sideband’ler modülatör frekansı ve modülasyon miktarının aynı anda etki ettiği nispeten karmaşık hesaplamalarla saptanabilir. Ancak genel bir kuraldan bahsetmek mümkündür: modülatörün

frekansının taşıyıcının frekansının tamsayı katları olduğu durumlarda (örn. taşıyıcı = 100Hz, modölatör = 400 Hz) ortaya çıkan sideband'ler taşıyıcının tamsayı katlarını, yanğı harmoniklerini oluşturacaktır. Bu sebeple taşıyıcı ile modölatör sinyallerin frekanslarının birbirinin tamsayı katı olduğu durumlarda elde edilen duyumlar çok daha doğıal olacaktır.



EREN
KARACAN

FMY-WEB.GITHUB.IO