C64 TÜRKİYE

SAYI : #$07 BİLGİ PAYLAŞTIKÇA ARTAR OCAK 2005

3. Boyut
C64 İçin PC Üzerinde Geliştirme
Futuristik Karakter Seti
Bold Karakter Seti
İntro Efektleri (Zıplayan Toplar)
İnterruptlar
Program Dökümleri

www.c64turkiye.com c64turkiye@yahoo.com hades@hi3s.org

3. BOYUT

HAZIRLAYAN : BİLGEM ÇAKIR
İLTİŞİM : nightlord@nightlord.dr2.net

Herkese selam. Üçüncü boyuttaki ilerleyişimize kaldığımız yerden devam ediyoruz. Geçen yazımızda, C64'te 3D efektleri yapmakla ilgili bazı felsefi soruları yanıtlamaya çalışmış ardından vektörlerin ne olduğu ve üzerlerinde toplama ve çıkarma işlemlerinin nasıl yapıldığını konuşmuştuk. Bu sayıda ise şu konulara değineceğiz:

.Vektör İşlemleri
..Vektörlerin boyu
..Vektör ile skaler çarpımı
..Vektörlerin nokta çarpımı (dot product)

.Vektörlerin 3D efektleri ile ne ilgisi var
..Objelerin vektör kullanılarak ifade edilmesi
..Dogrusal kenarlar ve yüzeyler
..Eğrisel kenarlar ve yüzeyler

.Vektör Dönüşümleri
..Dönüşüm kavramına giriş
..Yer değiştirme (transition)
..Dönme (rotation)
..Ölçekleme (scaling)

Dönüşümlerin birleştirilmesi

Bu yazıya başlamadan hemen önce gecen yazıyı bir kez daha okumanızı tavsiye ederim.

Vektör İşlemleri

Vektörlerde toplama ve çıkarma işlemlerinden bahsettikten sonra sıra çarpma işlemine geldi. Fakat vektörler söz konusu olduğunda çarpma işleminin üç değişik versiyonundan bahsedebiliriz.

Bir skaler ile bir vektörün çarpımı
İki vektörün nokta çarpımı (dot product)
İki vektörün çapraz çarpımı (cross product)

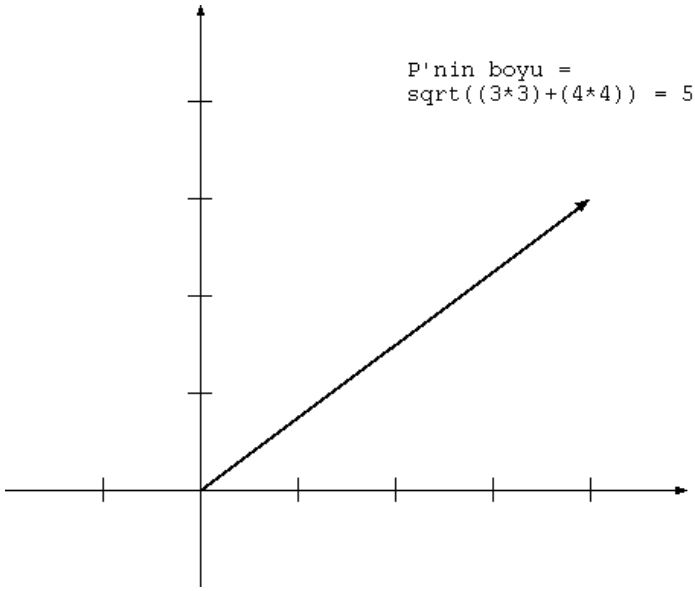
Çapraz çarpım konusuna bu yazıda değinmeyeceğiz. Vektörlerde çarpma işlemlerinden bahsederken ihtiyacımız olacak olan bir kavram daha var. Vektörlerin boyu. Bu konuyu kısaca açıkladıktan sonra skaler çarpıma geçebiliriz.

Vektörün Boyu

Vektörleri ilk başta tanımlarken bir büyüklük ve bir yön ifade ettiklerini söylemiştik. İşte bu büyüklüğü vektör koordinatlarından elde etmenin yolundan bahsetmenin zamanı geldi. Aslında bir vektörün boyunu bulmak için pisagor teoremini kullanmamız yeterlidir. P vektörünün boyunu [P] diye gösterirsek:

P(x1,y1) olsun

$$[P] = \sqrt{(x1^2 + y1^2)}$$



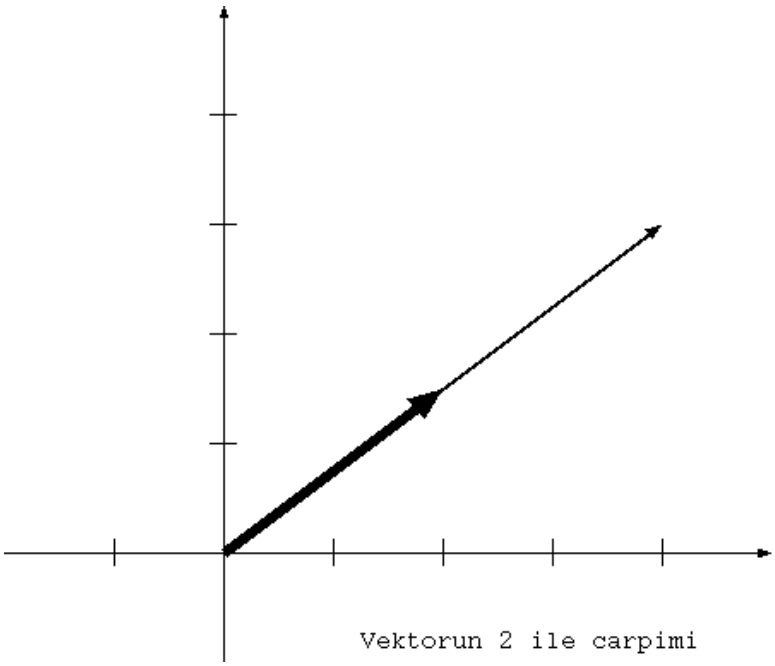
Vektör ile skaler çarpımı

Hatırlarsanız vektörler için çok boyutlu büyüklükler, skalerler için ise tek boyutlu büyüklükler demiştik. Bir skaler ile bir vektörün çarpımı tanımlı bir işlemdir ve sonucu da bir vektördür. Bütün yapmamız gereken skaler ile vektörün tüm bileşenlerini teker teker çarpıp sonuçlardan yeni bir vektör oluşturmaktır:

n skaler bir sayı olsun. P(x1,y1) vektörünü n ile çarparsak:

n * P(x1,y1) = (n*x1, n*y1)

n pozitif bir sayı ise bu yaptığımız çarpım, Vektörün yönünü değiştirmeden boyunu n oranında uzatır. n negatif ise yönü tam olarak tersine çevirip n'nin mutlak değeri oranında boy uzatılır. Uzatılır diyorsak da unutmayın n eğer 0 ile 1 arasında bir değere sahipse o zaman vektörün boyu kısalmış olur.



Vektörlerin nokta çarpımı

İki vektörün nokta çarpımı şöyle tanımlanmıştır:

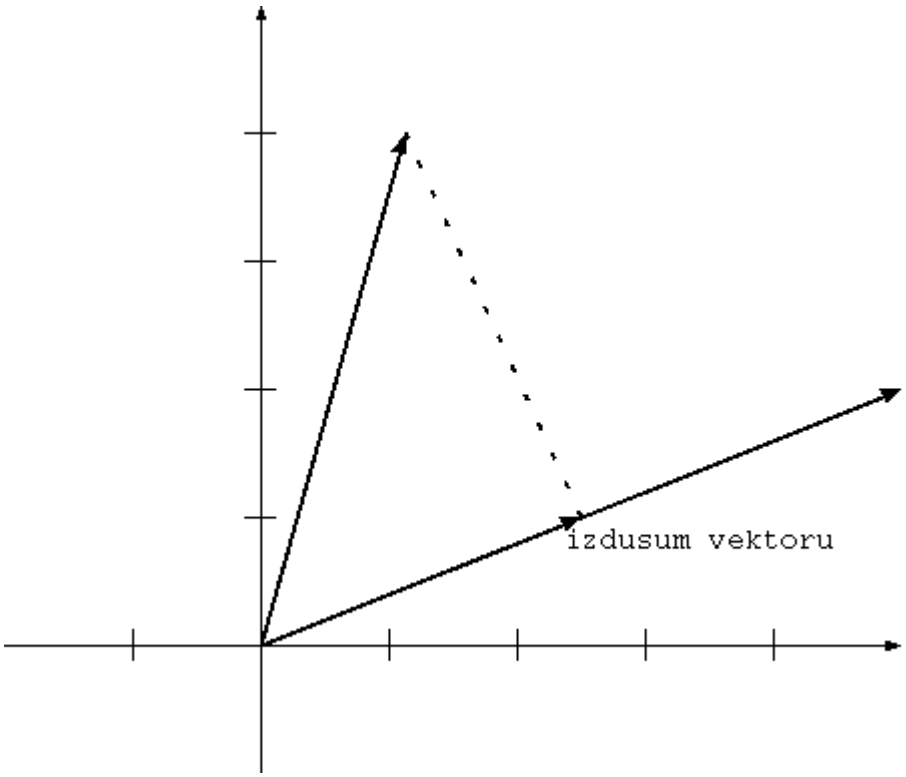
P(x1,y1) . Q(x2,y2) = (x1*x2) + (y1*y2)

Yani x koordinatları birbiri ile çarpılır. Aynı şekilde y koordinatları birbiri ile çarpılır. Bu iki çarpım toplanır. Yani nokta çarpımın sonucu bir skalerdir.

Bazı trigonometrik çıkarımlardan sonra bu işlemin aslında aşağıdaki gibi de ifade edilebildiğini görürüz.

P(x1,y1) . Q(x2,y2) = [P][Q]cos(a)

Burada a açısı P ve Q vektörlerinin arasındaki açıdır. [P] P'nin uzunluğudur (boyudur).Bu işlem geometrik olarak şöyle yorumlanabilir. [P]cos(a) aslında p vektörünün Q üzerine olan izdüşümünün boyudur. Bu boy ile Q'nun boyunun çarpımı, nokta çarpımını verir.



Örnek: (2,5) . (1,-2) = (2*1) + (5*(-2)) = 2 + (-10) = -8

Nokta çarpımın 3D efektlerdeki en önemli faydalarından biri iki vektör arasındaki açıyı bulmaya yaramasıdır. Bu açı da ışıklandırma efektlerinde işe yarar.

(x1*x2) + (y1*y2) = [P][Q]cos(a)

cos(a) = ((x1*x2) + (y1*y2)) / ([P][Q])

cos(a)'yı bildiğimiz için a'yı da bulabiliriz. Ama çoğu zaman zaten bize lazım olan a'nın kendisi değil cosinüs'üdür. Bunu ileride göreceksiniz.

Son olarak ufak bir gözlem yapmanızı isteyeceğim. İki vektörün aralarındaki açı 90 derece ise nokta çarpımları ne olur? cos(90)'ın değeri 0'dır. Yani vektörlerin boyları ne olursa olsun 0 ile çarpılacakları için aralarındaki açı dik olan iki vektörün nokta çarpımı her zaman 0 olacaktır.

Vektörler ile 3D efektlerin ne ilgisi var ?

Bu kadar çok vektörlerden bahsetmemize rağmen henüz sadece vektörler üzerinde bazı işlemlerin matematiksel olarak nasıl yapılacağını gördük. Bu işlemlerin ve vektörel ifadelerin ne işimize yarayacağını anlayabilirsek onları gereken yerlerde kullanabilecek durumdayız.

Objelerin vektörlerle ifade edilmesi

İşte kilit konu bu. Uzayda duran herhangi bir obje (kalem, kitap, ev, Hüsnü, Ayşe vs...) nasıl matematiksel olarak ifade edilebilir. Bir objenin şeklini matematiksel olarak ifade etmenin bir yolunu bulmalıyız ki o obje hakkındaki bilgileri bilgisayarda saklayabilelim ve üzerinde işlemler yapabilelim.

Her objeyi sonsuz sayıda noktalardan oluşuyor gibi düşünebiliriz. Yani sonsuz miktarda belleğimiz, sonsuz güçte bir cpu'muz olsa bir objeyi oluşturan bütün noktaları bir vektör olarak alıp her nokta için x ve y koordinatlarını belleğe saklayabilir ve üzerlerinde işlem yapabilirdik. Bu pratikte mümkün olmadığına göre bütün noktaları seçmeyeceğiz.

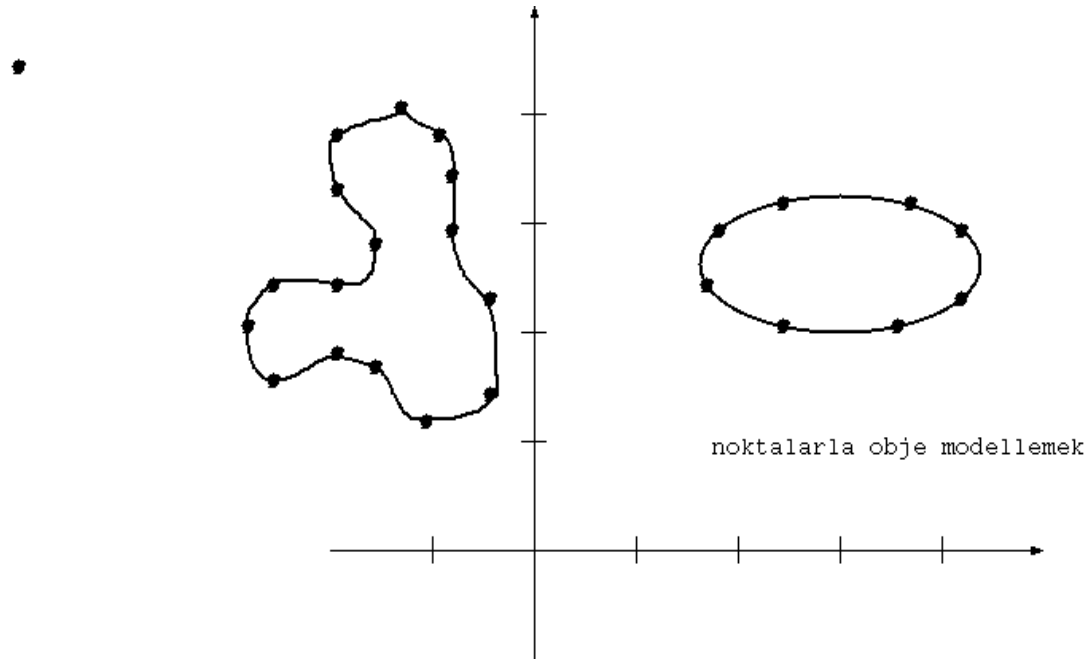
Eğer objenin sınırlarındaki noktaları seçip içini göz ardı edersek bayağı bir noktadan kurtulmuş oluruz. Yani artık objenin sadece dış yüzeylerindeki noktaları düşünelim. Maalesef hala sonsuz sayıda nokta demektir bu. iste bu noktada bu sonsuz sayıdaki dış noktadan sonlu sayıda nokta seçip objeyi yaklaşık bir şekilde modelleyeceğiz.

Modellemek

Modellemek bütün mühendislik dallarında kullanılan bir terimdir. Genel olarak gerçek dünyada var olan bir objeyi, olayı veya problemi matematiksel olarak ifade etmeye verilen addır. Çoğunlukla bu matematiksel ifadeye çevrim esnasında bazı bilgiler kaybedilir ve sisteme bir 'hata payı' dahil olur. Bu hata payının miktarı bilindiği müddetçe modelden verimli sonuçlar elde edilebilir. Mesela bir kalemin boyunu ölçseniz ve 20 cm deseniz bir modelleme yapmış olursunuz. Bu modelleme ile gerçek dünyada var olan 'kalemin boyu' kavramını matematiksel bir büyüklük olan '20 cm' ile modellemiş olursunuz. Cetvelinizdeki en dar çentik 1 mm ise siz kalemin boyunu 20.0 cm ile 20.1 cm arasında gördüğünüz için 20 cm dersiniz ama kalemin gerçek boyu belki de 20.024 cm. ama sizin elinizdeki imkanlarla bunu bilme şansınız yok. Peki modeliniz bu durumda hiç işe yaramaz mı? Mesela birisi dese ki “ben bu kalemden 4 tanesini uç uca koyacağım acaba şuradaki 1 metrelik masaya sığar mı?” Siz bu modeli kullanarak bu probleme cevap verebilirsiniz. Elinizdeki model kalemin 20 cm olduğunu ve 1 mm'lik hata payınız olduğunu söyler. Öyleyse 4 kalemin boylarında 4 kere maksimum hatayı goze alırsanız 4 kalemin uç uca 80 cm olacağını ve 4mm'lik hata payınız olduğunu söyleyebilirsiniz. Bu da 1 m'nin altında olduğu için bu kalem manyağı arkadaşınıza içinin rahat olmasını söyleyebilirsiniz.

Obje Modellemek

Böylece objenin dış yüzeylerinden seçeceğini sınırlı sayıdaki nokta ile o objeyi modelleyebilirsiniz. bu noktada objenin özelliklerine bakılarak nasıl bir modelleme yapılacağına ve ne tip noktaların seçileceğine karar verilir. Örneğin elinizde tam düzgün küp şeklinde bir obje varsa bunun 8 köşe noktasını seçerek modelleyebilirsiniz. Bu noktaların aralarında doğrusal kenarlar olduğunu kenarların oluşturduğu karelerin de doğrusal yüzeyler olduğunu söylersiniz. Aralardan daha fazla nokta da seçebilirsiniz. Ama doğrusal olduğunu bir kere söylediğiniz bir kenarın iki dış köşesini vermişseniz ortasından üçüncü bir noktayı daha vermeye gerek yoktur. Çünkü zaten o iki kenarın doğrusal olduğunu bir kere söylediniz. Bu sayede o aradaki her noktayı tanımlamış oldunuz. Bunun dışında seçilecek her ekstra nokta sisteme getirilen gereksiz bir yük olacaktır.



Doğrusal Kenarlar ve Yüzeyler.

Eğer obje böyle doğrusal kenar ve yüzeylere sahipse genelde 3 boyutlu jargonda MESH (meş diye okunur) olarak bilinen modelleme ile modellenir. Bu metotda obje, köşe noktaları ve köşeleri bağlayan kenarlar ve kenarların arasında kalan yüzeyler cinsinden ifade edilir. Unutulmaması gereken şey bu kenarların doğrusal, yüzeylerin düzlemsel olduğudur.

Eğri Kenar ve Yüzeyler

Bu konuya kısaca değineceğiz. bazen eğri kenarları olan organik canlı, kumaş vs gibi objeler farklı matematiksel ifadelerle modellenir. Çoğunlukla polinomik yaklaşık modelleme teknikleri kullanan bu sistemlerde (splines, NURBS vs) işlemler daha güçlü matematik CPU'lara ihtiyaç duyar. Yani kısa vadede C64 üzerinde kalkışmasanız iyi olur. Önce doğrusal sistemlerde kendinizi geliştirmeniz daha mantıklı olur.

Vektörel Dönüşümler

Dönüşüm nedir?

En genel tabiriyle dönüşüm eldeki bir vektöre bir takım işlemler uygulayıp yeni bir vektör elde etmemizi sağlayan bir operasyondur. Yani elimizde P vektörü varken bu vektöre M dönüşümünü uygularsak P' vektörü elde ederiz. Mesela örnek bir dönüşüm şöyle olabilir. P vektörünün bütün koordinatlarına 3 eklemek. Ya da P vektörünün x koordinatını 5 ile çarpıp y koordinatını 2 ile bölmek. Aklınıza gelen her dönüşüm, sonuçta aynı boyutlu başka bir vektör verdiği müddetçe geçerli bir dönüşümdür. (Hatta aynı boyutlu olmak zorunda bile değildir. Mesela perspektif dönüşüm 3 boyutlu bir vektörü iki boyutlu bir vektöre dönüştürür. Buna sonra değineceğiz)

Dönüşümler çoğu zaman tek bir vektöre uygulanmazlar. Birçok vektöre aynı anda uygulanırlar. Uzayda objeler köşe noktaları ile tanımlanabileceği için bir objeyi oluşturan bütün noktalara bir dönüşüm uygulanarak obje üzerinde bir dönüşüm yapılabilir. Bu dönüşüm objenin şeklinde bir değişiklik yapmayıp uzaydaki yerini veya duruşunu değiştirebilir. Ya da objenin şeklinde bir deformasyon meydana getirebilir işte dönüşümlerin 3D efektlerdeki kullanımı budur. donen küpler veya 3 boyutlu mekanlarda yapılan gezintiler hep aslında objeleri tanımlayan köşelere, o anda kameranın bulunduğu yer ve açıya göre bir takım dönüşümlerin uygulanması sonucu elde edilen yeni koordinatlara göre ekranda çizim yapılması ile gerçekleştirilir.

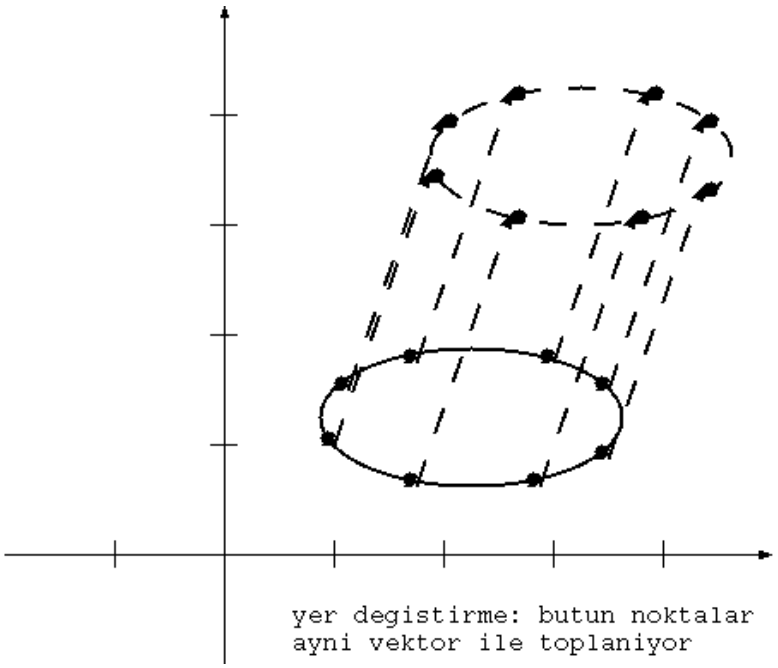
Yer Değiştirme

Bir noktanın koordinatlarına bir takım değerler eklediğimizde o noktanın yerini değiştirmiş oluruz. Bir objeyi modellerken kullandığımız bütün köşe noktalarına aynı dönüşümü uygularsak objeyi olduğu gibi uzayda hareket ettirmiş oluruz.

Genel olarak yer değiştirme hareketini de bir U(xh,yh) vektörü ile tanımlarsak, P(x1,y1) noktasını U vektörü kadar yer değiştirdiğimizde varacağımız noktaya Q(x2,y2) dersek,

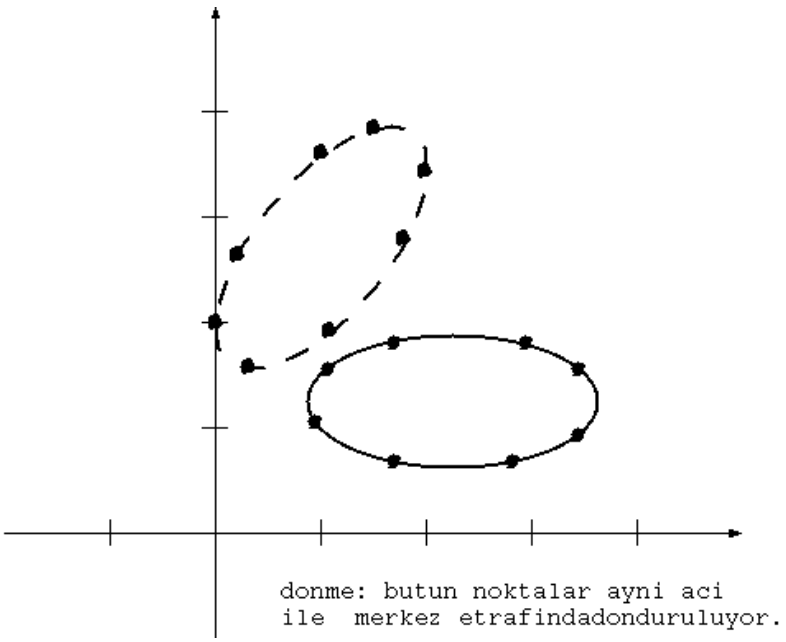
```
x2 = x1 + xh
y2 = y1 + yh
```

olur.



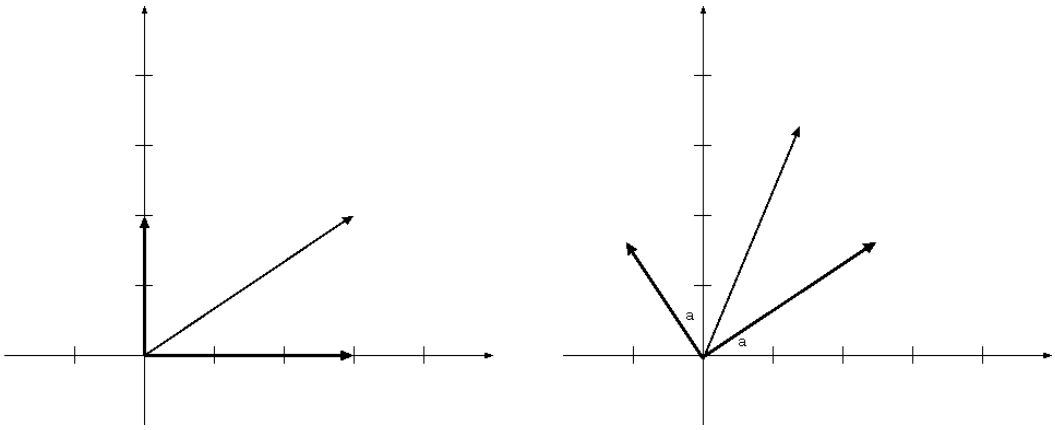
Dönme

Dönme işleminin nasıl olduğunu anlayabilmek için tek bir noktanın a açısı kadar origin etrafında döndürülmesini inceleyeceğiz.



Noktamız P(x1,y1) noktası olsun bu noktayı saatin ters yönünde a açısı kadar döndürdüğümüz zaman noktamız Q(x2,y2) koordinatlarına gelecek. Elimizde x1,x2 ve a varken x2 ve y2 yi nasıl bulacağız.

Bunun için gelin P vektörünü iki bileşene ayıralım. Px(x1,0) ve Py(0,y1). Dikkat ederseniz P'yi, Px'i ve Py'yi ne kadar döndürürsek döndürelim. P her zaman Px ve Py nin toplamına eşit olacak. Öyleyse biz Px ve Py'nin a açısı kadar döndürüldüğünde nereye geldiklerini bulursak (ki bunu bulmak kolay) sonra bu ikisini toplayıp Q(x2,y2) noktasını bulabiliriz



Px, a kadar dönünce x ekseni ile a derece açı yapan x1 uzunluğunda bir vektör olur. Trigonometriden hatırlarsak bu durumda Px'in x koordinatı (x1)cos(a) olur. Aynı şekilde y koordinatı ise (x1)sin(a) olur.

Py, a kadar döndüğünde y ekseninin sol tarafına doğru a açısı yapan y1 uzunluğunda bir vektör olur. onun koordinatları da sırası ile (-y1)sin(a) ve (y1)cos(a) olur.

Son olarak P'nin yeni yerini yani x2 ve y2 yi bulmak için Px ve Py nin aynı koordinatlarını toplarsak,

$$x2 = (x1)\cos(a) - (y1)\sin(a)$$
$$y2 = (x1)\sin(a) + (y1)\cos(a)$$

İşte bu bir noktayı tek eksende iki boyutlu döndürmek için gereken işlemdir. Bir objeyi oluşturan bütün noktalar için bu işlem tekrarlanırsa obje origin etrafında (ya da 3 boyutlu düşünürsek z ekseni etrafında) döndürülmüş olur.

Ölçekleme

Ölçekleme işlemi bir objeyi büyütüp küçültmeye yarar. Bir vektörün skaler ile çarpılması konusundan hatırlarsanız, bu işlem sonucu vektörlerin boyları uzayıp kısaltılabiliyordu. Eğer bir objeyi tanımlayan bütün noktaların vektörlerini aynı skaler ile çarparsak o objeyi o skaler oranında ölçeklemiş oluruz.

Genelde bütün noktaların x ve y koordinatları aynı skaler ile çarpılarak ölçeklenir bu durumda obje şeklinde herhangi bir deformasyona uğramadan büyür ve küçülür. Eğer x ve y koordinatları aynı oranda ölçeklenmezse objeler x veya y yönünde yassılaşırlar. Bu da bazen kullanılan bir efekttir. Ben bu efekti mist demosundaki wobbling vektör partında kullanmıştım.

Hatta objeyi oluşturan noktalar farklı farklı skalerler ile anime bir şekilde ölçeklenirse çok ilginc jelimsi objeler elde edilebilir ki bu tip efektler genelde çok pirim yapar. 3 boyut motorunuzu yazdıktan sonra böyle ufak oynamalarla çok ilginç ve orijinal efektler bulabilirsiniz.

Dönüşümlerin birleştirilmesi

Genelde dönüşümler tek tek değil birlikte gerçekleşirler. Yani bir obje uzayda aynı anda yer değiştirip kendi etrafında dönerken büyüyüp küçülüyor olabilir. Bir objeye olan dönüşümler bu durumda birleştirilip hesaplanabilmelidir.

Bu ihtiyacın doğduğu bir diğer popüler durum daha vardır. Kinematik. Örneğin kendi kolunuzu düşünün. Kolunuz omzunuzdan bir açı ile bir eksende dönerken, dirseğiniz başka bir açı ile döndüğünde eliniz ne kadar dönmüş ve yer değiştirmiş olur. Bu soruyu cevaplayabilmek için objelerin birbirlerine bir ebeveyn-çocuk hiyerarşisi ile bağlandığını düşünebiliriz. Yani bu örnekte gövdeniz, üst kolunuzun babası, üst kolunuz ön kolunuzun babası, ön kolunuz ise elinizin babası gibi ç.lç.mb babalar bir dönüşüm yaptıklarında bütün çocuk ve torunları bu dönüşümü geçirir. ama çocuk 10,

What is the matrix

Evet sayın okuyucular. Kısaca matrix matematiğinden de bahsederek gereken bütün temel matematik bilgisini tamamlamış olacağız. Bundan sonra da C64 üzerinde bu efektleri nasıl kodlayacağımıza geçeceğiz. Fakat bu sayı için yeterince şey öğrendiniz. Gelecek yazıda matrix'lerin tanımından devam edeceğim.

Tamamen matematik ağırlıklı olan bu yazıda anlamadığınız pek çok yer olabilir. Vektör işlemleri lise matematik seviyesinde olduğu için daha az problem olabilir ama dönüşümleri anlamak sizi daha çok zorlayabilir. Bu durumlar için forumlar veya mail yoluyla bana sorularınızı ulaştırabilirsiniz.

Gelecek yazıda görüşmek üzere...

C64 İÇİN PC ÜZERİNDE GELİŞTİRME

HAZIRLAYAN : BİLGEM ÇAKIR
İLTİŞİM : nightlord@nightlord.dr2.net

Merhaba. Bu yazı dizisinde PC kullanarak C64 için nasıl program geliştirebileceğinize dair bilgiler vermeye çalışacağım. Böylece o ruhsuz PC'niz de faydalı bir iş için kullanılmış olacak ☺

Öncelikle birkaç şeyi tanımlayalım. Bir PC kullanarak başka bir bilgisayar sistemi için yazılım geliştirmeye çapraz geliştirme (cross-development) adı verilir. Bu yalnızca C64 için yapılan bir şey değildir. Uzun yıllardır gömülü (embedded) sistem adı verilen bütün cihazlar (tv'ler, dvd'ler, çamaşır makineleri, fabrika robotları vs) bu şekilde geliştirilmektedir. Böyle yapılan geliştirmede PC'ye host platform, yazılımın aslında çalışacağı platforma da hedef platform denir.

Bunun çok basit sebepleri vardır. Genelde hedef platformlar RAM ve CPU gücü bakımından daha zayıftırlar. Bu yüzden onlar üzerinde janjanlı kod editörleri ve geliştirme araçları çalıştırmak mümkün olmayabilir. Çoğu zaman Hard diskleri de olmadığı için kaynak kodları saklamak ve düzenlemek daha zordur.

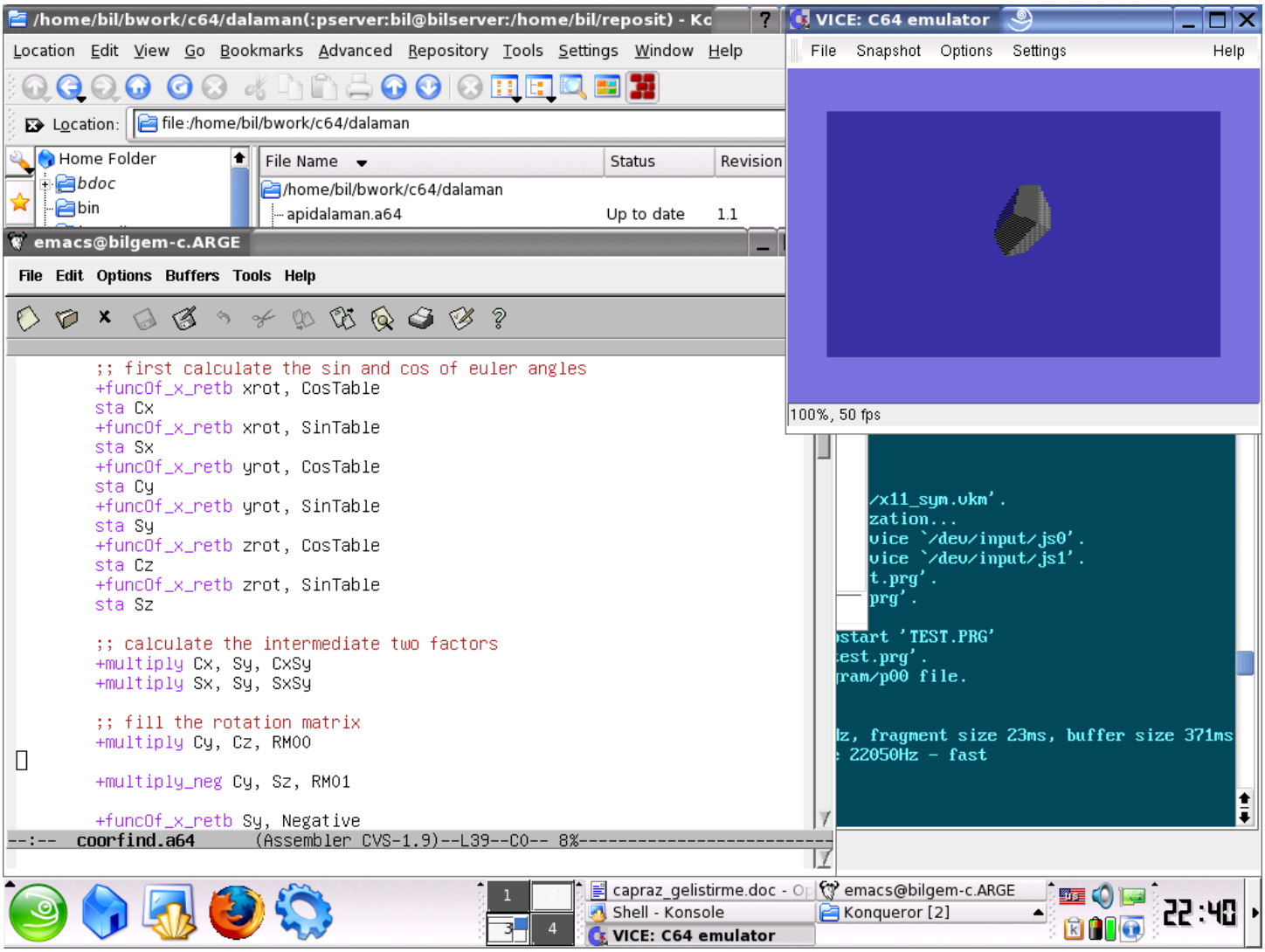
C64'ü özellikle düşünürsek 5,25 disketlerle kaynak kodları saklamak nispeten daha güvensiz olabilir. PC'de yazdığınız kodları defalarca kopyalar CD'lere yazar veya CVS gibi versiyon yönetim sistemlerinde saklayabilirsiniz. Bu kaynak kodlarda sonsuz uzunlukta değişken ve etiket isimleri kullanabilir kod içine istediğiniz kadar yorum yazabilirsiniz.

Bütün bu avantajların yanında bence PC üstünde geliştirmenin gerçekten çok zaman ve efor kazandırdığı bir durum vardır. Çok parçalı demolarda linking dediğimiz süreç. Bu süreçte çeşitli efektler içeren demo partlarını birleştirerek, geçişlerin zamanlaması ve

müzik senkronizasyonu ile uğraşılır. Bu esnada sürekli olarak efekt partlarındaki delay zamanları ile oynanıp müzik senkronizasyonu ile oynanır. Demo disketi tekrar tekrar silinip yeniden dosyalar kopyalanır. Ardından demo çalıştırılıp üzerinde son uğraştığınız part gelene kadar izlenir (ki bu 3-5 dakika alabilir). Ve tam istediğiniz yeri geldiğinde ekranı flaş yaptırdığınız anın müzikten 1 saniye geç kaldığını görürsünüz. bütün bunları tekrarlarsınız ve başka bir yerin zamanlaması bozulur. İşte bu korkunç emekli süreç PC üstünde make, crossassemblerlar, vice emülatörü ve bu emülatörün warp modu sayesinde çok kolaylaştırılabilir. Bunların nasıl yapıldığına değineceğiz.

C64 için PC'de çalışan pekçok emülatör, crossassambler, gibi araç mevcuttur. Hangisini seçeceğiniz, kişisel tercihlere kalmıştır. Ayrıca kullandığınız işletim sistemi de bunda etkili olabilir. Ben bu yazıda kendi kullandığım yapıyı tanıtacağım. Ben normalde işte ve evde linux pc kullandığım için benim kullandığım araçlar linux uyumlu araçlar ancak aynı araçlar windows'da da mevcut.

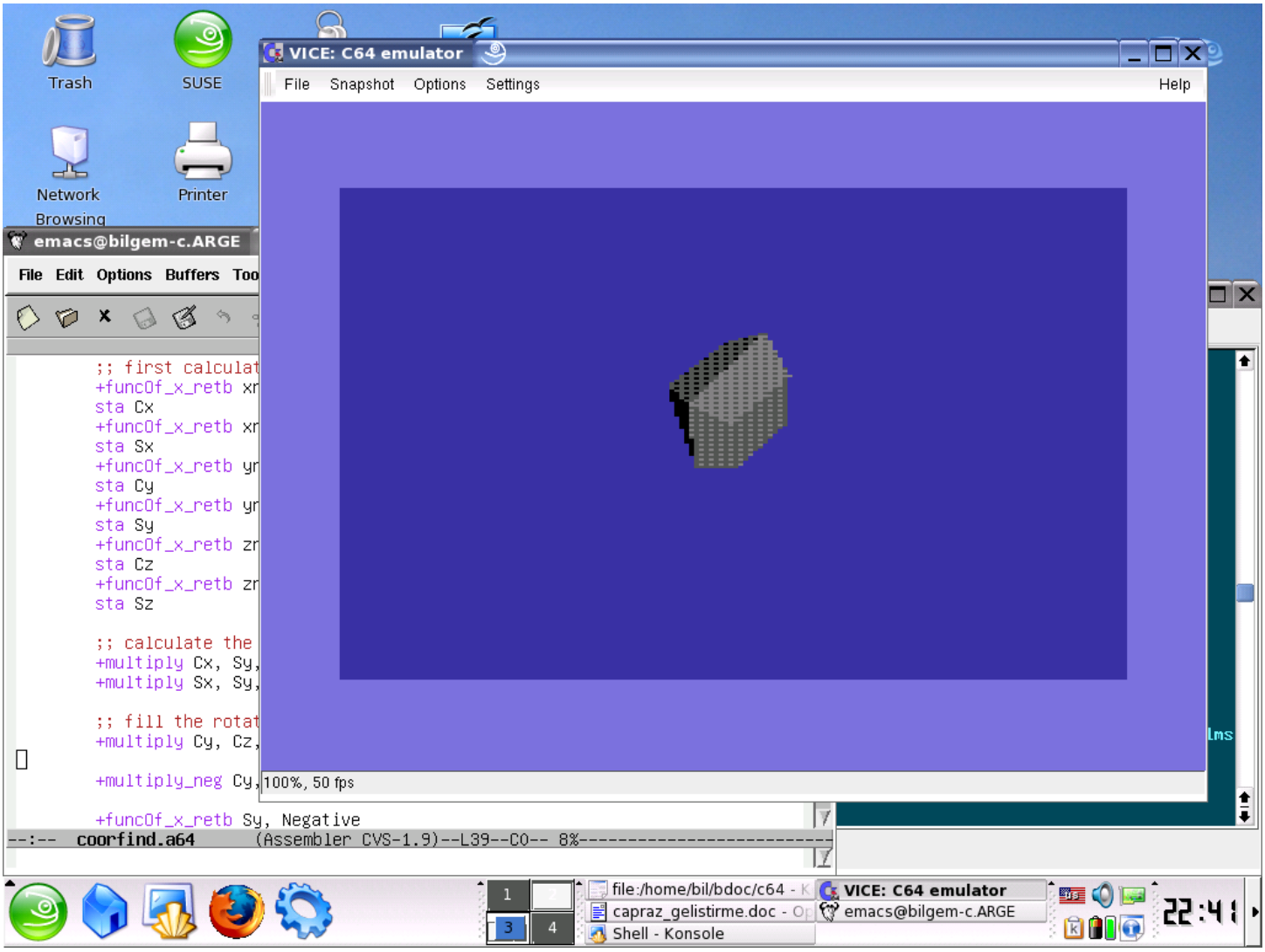
Bu noktada programcılık hakkında kısa bir felsefik paragraf yazayım. Her ne amaçla hangi platforma yönelik program yazıyor olursanız olun, eğer bugüne kadar yapmadıysanız linux kullanmayı denemenizi öneririm. Linux içinde hali hazırda bedava olarak gelen o kadar çok yazılım mühendisliği ve geliştirme aracı var ki, bunların yaptıkları işleri yapabilen windows karşılıklarını aradığınızda hepsini bulamayacağınız gibi bulabildiklerinize da yüzbinlerce dolar ödemek zorunda kalabilirsiniz. hem legal hem kuvvetli hem de bedava olan bu araçlar sayesinde linux'ta kod geliştirmeye alıştıktan sonra başka sistem istemeyeceksiniz. Maalesef demoscene gibi özgürlük, bedava sanat eserleri üretmek gibi kavramlarla uğraşan insanların dünyasında microsoft teknolojilerinin neden tek alternatifmiş gibi görüldüğüne inanmak zor. Tabii linux için bir öğrenme süreci geçireceğinizi de bilin ama bunun ödülünü fazlasıyla alacaksınız.



Neyse C64'e dönelim. Araçları tanıtmadan önce son olarak bir konuya daha değineyim. Çapraz geliştirme derken sadece kod yazmaktan bahsetmiyoruz. PC de var olan toolları kullanarak C64 için grafik ve müzik de geliştirebilirsiniz. Fakat özellikle grafik ve ses konusunda orijinal donanımın üstündeki havanın aynısını PC yaratamamaktadır. Bilhassa PC'de yaptığınız müzikler C64'e atınca çok farklı sesler çıkarabilir. Bizzat başıma geldi onun için söylüyorum. SID çipinin filtreleri PC'de emüle edildiğinden çok farklı sesler çıkarıyor. O yüzden grafik ve müzik çalışmalarınızı gerçek C64 ile yapmanızı tavsiye ederim.

VICE

Önce emülatör. Benim gözlemlediğim kadarıyla şu an dünyada en çok beğenilen ve kullanılan iki emülatör var: CCS ve VICE. Bunlardan ikisi de son derece iyi ve sorunsuz emülatörlerdir. Bugüne kadar hassas zamanlamalı VIC raster efektlerinden 3D efektlere kadar yaptığım ve izlediğim hiçbir efektte iki emülatörde de gerçek C64'ten farklı bir davranış görmedim. Sadece iki yıl önce CCS ile Fairlight'ın Anyone demosunu izlerken loader takılmıştı. Yani 1541 emülasyonunda bir hata vardı. Yeni versiyonlarda düzelmiş olabilir. Benim bilmediğim ya da rastlamadığım başka hatalar olabilir. Ama genel olarak ikisini de gönül rahatlığıyla kullanabilirsiniz. Linux desteği olan VICE olduğu için benim tercihim ondan yana oldu.



Genel olarak VICE kullanımı ile ilgili detaylı bilgi vermeyeceğim. Bunun için Ophruque'un yazdığı Türkçe çok güzel bir VICE kullanma klavuzu var buna bakabilirsiniz. Ben daha çok VICE ile debug yaparken kullanacağınız makine dili monitörüne değineceğim. Ama biraz sonra...

ACME

Önce crossassembler nedir onu söyleyeyim. Crossassembler, host platformda yazdığınız assembler dilindeki kaynak kodu hedef platformda çalışacak bir binary dosyaya dönüştüren araçtır.
Assembler programlarının genel ozellikleri olan etiketler ve macrolar ACME'nin de en işe yarayan özellikleri arasında. ACME'nin önemli bir özelliği ise çok hızlı çalışması.

Kod editörü: EMACS

PC'de kod yazarken önemli bir avantajınız, çok fazla değişik kod editörü seçme imkanınız olması. Kod editörü içinde herhangi bir dilde program yazdığınız araçlardır. Windows Not defteri yada Microsoft Visual Studio da kod editörü olarak kullanılabilir. Genelde Bir kod editörünün içinden kodları derleyip çalıştırabilmenize ve hatta debug edebilmenize izin verecek şekilde yapılır. Bunlara entegre geliştirme çevreleri (integrated development environment) anlamına gelen IDE adı verilir. Pekçok programcı aslında günümüzde IDE'leri tercih ediyor. Benim gibi dinazorlar ve UNIX tabanlı programcılar ise editor compiler debugger vs için ayrı ayrı araçları tercih ederler. Bu araçlarda ki avantaj şudur. Aynı kod editörü ile hem C64 hem PC hem de başka platformlar için kod yazabilirsiniz. Yani compiler/assembler veya debugger değiştirirken alıştığınız kod editörünü kullanmaya devam edebilirsiniz.

Kod editörü gerçekten çok kritik bir konu değildir. Herkes genelde rahat ettiği bir editör bulur ve onunla devam eder. Ben üç yıl önce evde ve işte EMACS kullanmaya başladım EMACS şu meşhur özgür yazılım hareketini 80'lerde başlatan FSF (Free Software Foundation) tarafından yürütülen GNU projesinin editörüdür. Avrupa ve Amerika'daki pek çok yazılımcı tarafından VI editörü ile birlikte dünyanın en iyi editörü kabul edilir.

VI ve EMACS görünüşte çok sade yazılımlardır. Fakat her ikisinin de temel prensibi kullanıcının elini normal daktilo konumunda tuttuğu zaman elini klavyeden hiç kaldırmadan pek çok işi yapabilmesidir. Genelde pek çok diğer janjanlı editördeki gibi mouse ile klavye arasında elinizi götürüp getirerek zaman kaybetmeniz gerekmez. İkinci önemli özellik de EMACS in pek çok bölümünün lisp adı verilen bir dille yazılmış olmasıdır. Bu dille çok çeşitli küçük rutinler yazıp editöre eklemeniz mümkündür. Yani herhangi bir istediğiniz özelliği genelde 20-50 satır lisp kodu yazarak EMACS'e kazandırabilirsiniz.

Eğer EMACS kullanmaya karar verirseniz tutorialleri takip ederek tuş kullanımını ve kısayolları iyice öğrenmeye çalışın. Bu başta zor gelecek ve biraz zaman alacaktır. Ama alıştıktan sonra size çok rahat gelebilir.

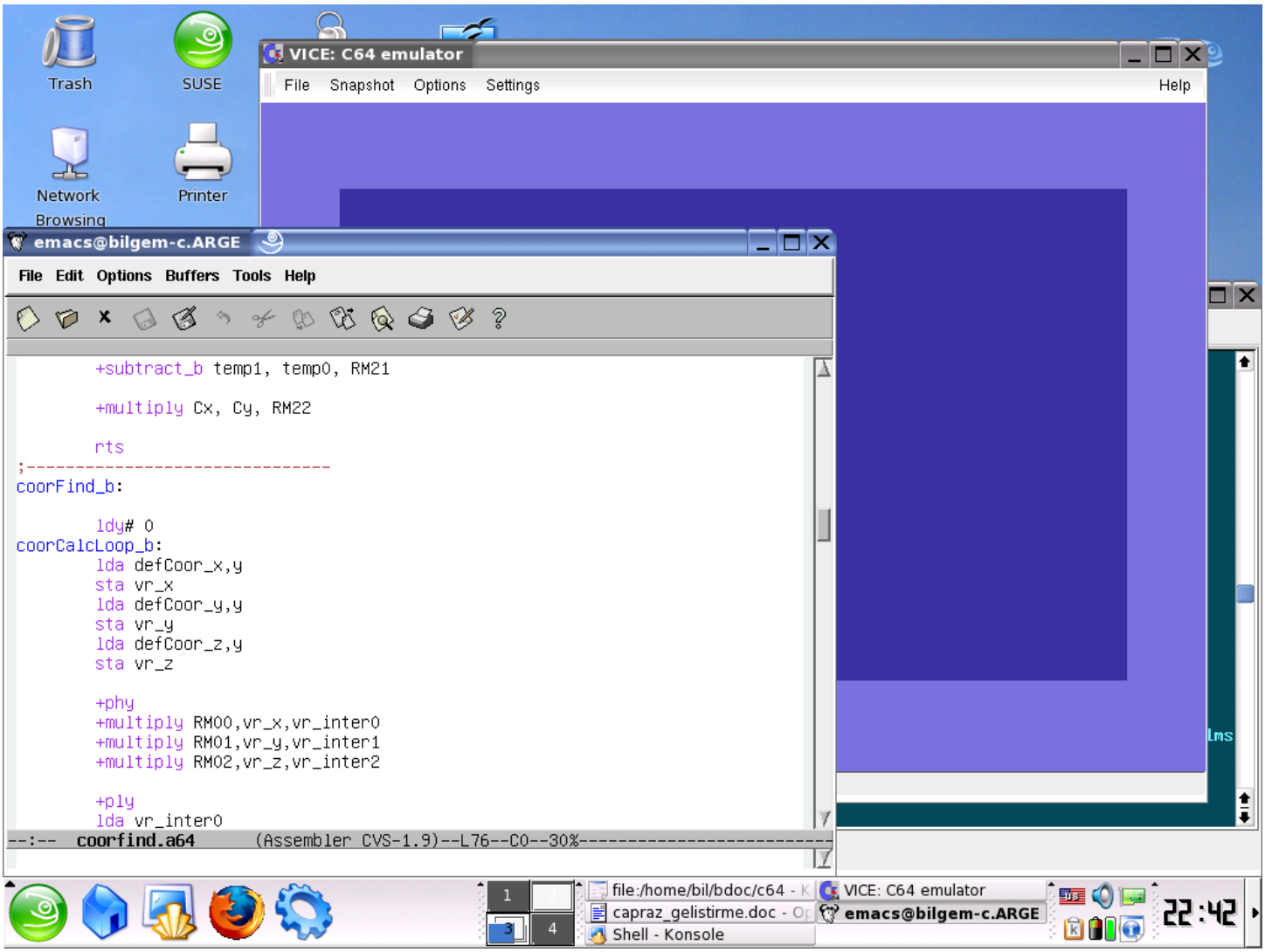
Geliştirme döngüsü

Kod geliştirirken kullandığımız üç ana araçtan bahsettik:

1- kod editörü: EMACS
2- assembler: ACME
3- debugger/emülatör: VICE

Normalde kod geliştirirken sürekli bu üç ana araç arasında döneceksiniz. Öncelikle editör içinde kod yazacak ve bir dosyaya kaydedeceksiniz. Ardından bu dosyayı ACME ile assemble edeceksiniz. Eğer bir takım yazım hataları yaptıysanız ACME size hangi dosyanın hangi satırında hata yaptığınızı söyleyen hata mesajları çıkaracak. Ardından bunları tek tek editörde yeniden düzelteceksiniz. En sonunda ACME bir hata vermeden kaynak kodlarınızı assemble edecek ve bir C64 dosyası oluşturacak. Daha sonra bu dosyayı VICE emülatörü ile çalıştıracaksınız. Büyük olasılıkla çalışmayacak. VICE monitör ile bir süre debug edip hataları bulacaksınız. Bu hataları EMACS ile düzeltip yeniden ACME ve yeniden VICE çalıştıracaksınız. Ta ki yazdığınız kod parçası istediğiniz gibi çalışana kadar.

Genelde biraz büyükçe projeleri önce kafanızda çalıştırılması gereken alt parçalara böldüğünüz için VICE üstünde parçaları tek tek çalıştırıp memnun kalınca bunları birleştireceksiniz. Birleştirme sonucu çıkan hatalar da ayıklandığı zaman sonuca ulaşmış olacaksınız. C64 üstünde istediğiniz gibi çalışan bir ürün.



Gelecek Yazı

Bu ay burada keseceğim. Gelecek yazıda ACME'nin özel komutları ve bilhassa da Macro'ların kullanımına değineceğim. Ayrıca ACME, VICE ve EMACS dışında yardımcı olarak kullandığım iki araçtan daha bahsedeceğim: PUCRUNCH ve MAKE.

O zamana kadar hoşça kalın ...

(Editörün notu : Eğer LİNUX yerine WİNDOWS 98 kullanıyorsanız kod editörü olarak wordpad, assembler derleyici olarak C64ASM V1.1 ve emülatör olarak VICE kullanabilirsiniz.)

PROGRAM BÖLÜMÜ

Kısa bir aradan sonra kaldığımız yerden devam ediyoruz. Kısa programlarla karakter setimi değiştirerek açılış yapıyoruz. Nasıl olduğunu merak ediyorsanız ekran görüntüsüne bakabilirsiniz. İkinci programımızda ise karakterler biraz genişliyor. Aslında bu rutinler ile elde ettiğimiz karakterlerden daha güzellerini "FONT EDİTÖR" dediğimiz programlarla yapabilirsiniz. Karakterleri bir tarafa bırakalım ve devam edelim. İntro teknikleri bölümünde ise sprite'ları hareket ettirerek ekranımızı güzelleştiriyoruz.

FUTURİSTİK KARAKTER SETİ PROGRAMI
KARAKTER SET NO:5

HAZIRLAYAN : İSMAİL ŞAHİN
İLETİŞİM : HADES@HI3S.ORG

(C64ASM V1.1 ile compile edilecek şekilde yazılmıştır.)

```
                *=$0801

                .word nextline
                .word 2004        ; 2003
                .byte $9e         ; SYS
                .text "2061"      ; 2061
                .byte 0           ; komutu
nextline        .word 0

                sei
                lda   #$33
                sta   $01

                lda   #$d0
                ldx   #$30
                ldy   #$00
                sty   $fb
                sta   $fc
                sty   $fd
                stx   $fe

                ldx   #$10
transfer        lda   ($fb),y
                sta   ($fd),y
                iny
                bne   transfer
                inc   $fc
                inc   $fe
                dex
                bne   transfer
                lda   #$37
                sta   $01
                cli
                lda   #$1c
                sta   $d018

new_fonts       lda   #$30
                ldy   #$00
                sty   $fb
                sta   $fc
```

```
                ldx   #$10
                stx   $fd

loop1           iny
                iny
                iny
                ldx   #$04
loop0           lda   ($fb),y
                lsr
                ora   ($fb),y
                sta   ($fb),y
                iny
                dex
                bne   loop0
                iny
                bne   loop1
                inc   $fc
                dec   $fe
                bne   loop1
                rts

                .end
```

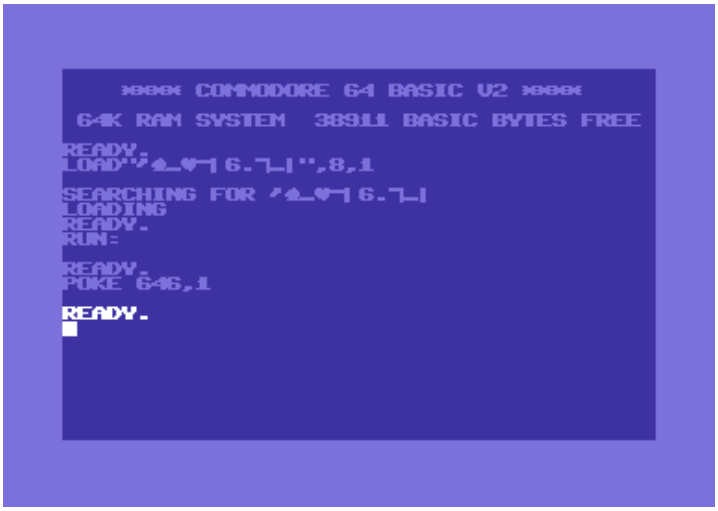
BOLD KARAKTER SETİ PROGRAMI
KARAKTER SET NO:6

HAZIRLAYAN : İSMAİL ŞAHİN
İLETİŞİM : HADES@HI3S.ORG

(C64ASM V1.1 ile compile edilecek şekilde yazılmıştır.)

```
                *=$0801

                .word nextline
                .word 2004        ; 2003
                .byte $9e         ; SYS
                .text "2061"      ; 2061
                .byte 0           ; komutu
nextline        .word 0

                sei
                lda   #$33
                sta   $01

                lda   #$d0
                ldx   #$30
                ldy   #$00
                sty   $fb
                sta   $fc
                sty   $fd
                stx   $fe

                ldx   #$10
transfer        lda   ($fb),y
                sta   ($fd),y
                iny
                bne   transfer
                inc   $fc
```

```
                inc   $fe
                dex
                bne   transfer

                lda   #$37
                sta   $01
                cli
                lda   #$1c
                sta   $d018

new_fonts       lda   #$30
                ldy   #$00
                sty   $fb
                sta   $fc

                ldx   #$10
loop0           lda   ($fb),y
                and   #$f0
                pha
                asl
                sta   $fd
                pla
                ora   $fd
                sta   $fe
                lda   ($fb),y
                and   #$0f
                pha
                lsr
                sta   $fd
                pla
                ora   $fd
                ora   $fe
                sta   ($fb),y
                iny
                bne   loop0
                inc   $fc
                dex
                bne   loop0
                rts

                .end
```



5 Numaralı karakter setinin ekran görüntüsü

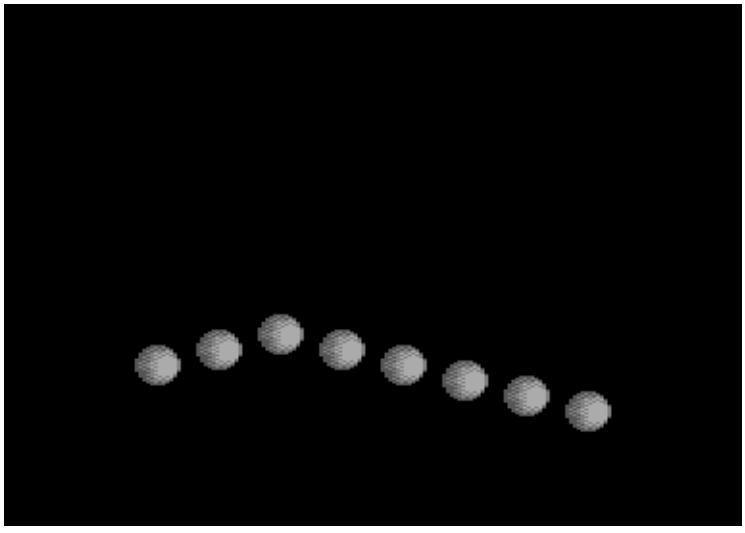
6 numaralı karakter setinin ekran görüntüsü

İNTRO EFEKTLERİ

Herkese merhaba. İlk olarak geçen sayıda intro teknikleri olarak başladığımız yeni yazı dizisinin adını "intro efektleri" olarak değiştirdiğimizi belirtelim. İnternette yapacağınız kısa bir araştırma ile yüzlerce intro bulabilir ve bu introlarda birbirinden güzel efektler görebilirsiniz. Bu efektleri yapabilmek için bir yerden başlamanız gerekmektedir. İşte bu nedenle "intro efektleri" bölümü sizin için başlangıç olacaktır. Benim bu sayfalarda vereceğim efekt örnekleri sadece fikir amaçlıdır ve daha güzel ve orijinal efektleri sizlerde yapabilirsiniz. Aslında bu giriş yazısını geçen sayıda yazmam gerekiyordu. Neyse lafı uzatmadan bu sayıdaki efektten kısaca bahsedeyim. Efektimiz "bouncing balls" denilen zıplayan toplar efektidir. Efekte ait ekran görüntüsünü program listesinin sonunda görebilirsiniz.

```
                *=$0801
                .word nextline
                .word 2004          ; 2004
                .byte $9e           ; SYS
                .text "2061"        ; 2061
                .byte 0             ; komutu
nextline        .word 0

start           sei                 ;İnterruptları engelle
                jsr   $e544         ;Ekranı sil.(geri dönüşte X registerinde 1 vardır)
                dex                 ;X registerini 1 azalt. (X registeri 0 olur)
                stx   $d020         ;Ekran dış rengi siyah
                stx   $d021         ;Ekran iç rengi siyah
loop0           lda   #(sdata/64)   ;sprite datasının yerini
                sta   $07f8,x       ;sprite pointerlerine yaz
                lda   koor,x        ;0-3 arası sprite'ların koordinat değerlerini
                sta   $d000,x       ;VIC'teki ilgili koordinat registerlerine kopyala
                lda   koor+8,x      ;4-7 arası sprite'ların koordinat değerlerini
                sta   $d000+8,x     ;VIC'teki ilgili koordinat registerlerine kopyala
                lda   #$0f          ;#$0f (açık gri) değerini
                sta   $d027,x       ;sprite renk registerlerine yaz
                inx                 ;X registerini 1 arttır.
                cpx   #$08          ;8 oldumu ?
                bne   loop0         ;olmadıysa işlemleri tekrarla
                lda   #$ff          ;Bütün spriteları
                sta   $d015         ;göster ve
                sta   $d01c         ;multicolor yap
                ldx   #$0b          ;koyu gri renk kodu
                ldy   #$0c          ;gri renk kodu
                stx   $d025         ;multicolor1 renk registeri
                sty   $d026         ;multicolor2 renk registeri
                lda   #<irq         ;interrupt rutinimizin başlangıç adresi LO byte'ı
                ldx   #>irq         ;interrupt rutinimizin başlangıç adresi HI byte'ı
                sta   $0314         ;interrupt vektörü LO byte'ına yaz
                stx   $0315         ;interrupt vektörü HI byte'ına yaz
                ldx   #$01          ;raster line interruptı yapacağımızı
                stx   $d01a         ;VIC'e bildiriyoruz
                ldy   #$80          ;256'dan büyük X koordinatına sahip sprite için
                sty   $d010         ;EXB'yi aktif hale getiriyoruz
                cli                 ;interruptları serbest bırakıyoruz
                jmp   *             ;sonsuz döngüye gir.

irq             inc   $d019         ;bir sonraki interrupta izin ver
wait            lda   $d012         ;raster line adresini oku
                bne   wait          ;sıfır değilse bekle. Sıfırsa devam et
                lda   delay         ;Gecikme sayacını oku
                bne   exit          ;sıfır değilse çık.
                lda   #$0a          ;sıfır olduysa başlangıç değerini
```

```
                sta   delay       ;gecikme sayacına yaz.
                ldx   cnt         ;tablo sayacının al
                cpx   #$20        ;$20 (32) oldumu?
                bne   cont        ;olmadıysa sıçra.
                ldx   #$08        ;evet olmuş. Tablo sayacının başlangıç
                stx   cnt         ;değerini tablo sayacına yaz.
cont            ldy   #$00        ;Y=0 sprite koordinat sayacı
loop1           lda   table,x     ;tablo sayacına göre tablodan o anki byte'ı oku
                sta   $d001,y     ;o anki sprite'in Y koordinat adresine yaz.
                inx               ;X registerini 1 arttır
                iny               ;Y registerini 1 arttır
                iny               ;Y registerini 1 arttır
                cpy   #$10        ;Y $10 (16) oldumu?
                bne   loop1       ;Olmadıysa işlemleri tekrarla
                inc   cnt         ;Tablo sayacını 1 arttır
exit            dec   delay       ;gecikme sayacını 1 azalt
                jmp   $febc       ;Kontrolu işletim sistemine devret

cnt             .byte 0           ;tablo sayacı
delay           .byte 0           ;gecikme sayacı

                *=$08c0           ;spriteların şekil datalarının başlangıç adresi

sdata           .byte $00,$55,$00,$01,$77,$40,$05,$dd
                .byte $d0,$07,$7f,$b0,$15,$ee,$e4,$17
                .byte $fb,$bc,$1d,$ee,$ac,$57,$fa,$a9
                .byte $5d,$ea,$ab,$77,$ba,$ab,$5f,$ea
                .byte $ab,$77,$ba,$ab,$5d,$ea,$ab,$57
                .byte $fa,$a9,$15,$ea,$a8,$17,$fa,$ac
                .byte $15,$ee,$e4,$07,$7b,$b0,$05,$ff
                .byte $d0,$01,$77,$40,$00,$55,$00,$00

koor            .byte $3c,$b0,$5c,$b0,$7c,$b0,$9c,$b0    ;spriteların başlangıç
                .byte $bc,$b0,$dc,$b0,$fc,$b0,$1c,$b0    ;koordinat değerleri

table           .byte $b0,$b0,$b0,$b0,$b0,$b0,$b0,$b0    ;tablo sayacına göre
                .byte $b8,$c0,$c8,$d0,$d8,$e0,$d8,$d0    ;spriteların Y koordinat
                .byte $c8,$c0,$b8,$b0,$b8,$c0,$c8,$d0    ;değerleri
                .byte $d8,$e0,$d8,$d0,$c8,$c0,$b8,$b0
                .byte $b8,$c0,$c8,$d0,$d8,$e0,$d8,$d0

                .end              ;programın sonu
```



Zıplayan toplarımızın ekran görüntüsü.

İNTERRUPTLAR

İki sayıdır intro efektleri bölümünde interruptları kullanarak bir şeyler yapıyoruz. Aslında "interrupt" konusunu intro efektlerine başlamadan önce anlatmam gerekiyordu. Ama yinede geç kalmış sayılmayız ☺
Öncelikle "interrupt" kelimesi ne demektir, C64'te ne işe yarar vs. soruların cevabını verelim. Kelime anlamı olarak "interrupt" "kesmek, akışını durdurmak, yarıda kesmek, düzenini bozmak" demektir. Bu durumda biz "interrupt" ile C64'te neyi kesiyoruz, neyin akışını durduruyoruz buna bir göz atalım. C64'ü açıp karşınıza gelen ilk ekrana bakın. İlk gözünüze çarpan yanıp sönen bir kursör vardır. Oysa C64'ün içinde bundan çok daha fazlası yapılmaktadır.

C64'te sürekli çalışan bir program vardır ve bunun adı KERNAL dediğimiz işletim sistemidir. Bu işletim sistemi sürekli olarak tuşları okur, kursörü yakıp söndürür, sistem saatini günceller vs..

Hazır başlamışken biraz daha detaya inelim ve mikroişlemciden başlayalım. C64'e ilk enerji verdiğimizde aslında mikroişlemci kendi içerisinde bir şeyler yapmaktadır. Tamamen hardware kaynaklı olan bu işlemler başlamadan önce işlemci besleme geriliminin belli bir seviyeye çıkması için bekler. Ama bu bekleme çok kısa olduğu için biz fark etmeyiz. Daha sonra işlemcinin tasarım aşamasında belirlenen ve ilk enerji verildiğinde hangi adresler-den itibaren çalışmaya başlayacağı bilgisi vardır. Bu bilgiler yani ilk çalışma adresleri 6510 için vektör adres olarak tasarlanmıştır ve adreslenebilen bellek aralığının son 6 adresidir. Hardware vektör denilen bu adresler aşağıda verilmiştir.

VEKTÖR			VEKTÖR ADI
$FFFA	43 FE	$FE43	NMI VECTOR
$FFFC	E2 FC	$FCE2	RESET VECTOR
$FFFE	48 FF	$FF48	IRQ VECTOR

Yukarıdaki tabloda işlemci ilk enerji verildiğinde NMI (Non maskable interrupt-engellenemez interrupt) işlemleri için $FFFA ve $FFFB adreslerinde tutulan sayıların ($43 ve $FE) gösterdiği adrese dolaylı sıçrama (indirect jump) yapar ve çalışmaya $FE43 adresinden itibaren devam eder. Aynı şekilde RESET için $FCE2 adresinden, IRQ (Interrupt ReQuest) için $FF48 adresinden devam eder. Bu arada belirteyim, RESET işlemi en yüksek önceliğe sahiptir yani o anda işlemci ne iş yapıyor olursa olsun hatta kilitlenmiş olsun hardware olarak –mesela bir reset buton ile- bir RESET işareti geldiği anda her şeyi yeniden başlatır. Veya işlemcinin NMI, RESET ve IRQ girişlerine aynı anda işaret uygulayın önce RESET işlemleri yapılacaktır.

Vektör adresleri yani $FFFA----$FFFF sabittir. Sadece bu adreslerdeki sayılar değişebilir. Diyelim ki C64 için yeni bir işletim sistemi tasarladınız. Yapmanız gereken sadece bu vektör adreslerdeki sayıları kendi NMI-RESET-IRQ rutinlerini gösterecek şekilde değiştirmek olacaktır. Neyse konuyu fazla dağıtmadan devam edelim.

IRQ vektörünün gösterdiği adresten itibaren ($FF48) rutini incelediğimizde $FF58 adresinde JMP ($0314) komutunu görürüz. İşlemci interrupt işlemleri sırasında bu adrese geldiğinde $0314 ve $0315 adreslerinde tutulan sayıların gösterdiği adresten itibaren çalışmaya devam edeceğini anlar. Bu arada $0314 ve $0315 adresleri RAM'da olup bu adreslerde tutulan sayılar daha önce RESET rutininin ilk çalışması sırasında yerleştirilmiştir

$0314 adresinde $31, $0315 adresinde ise $EA sayıları bulunmaktadır ve JMP ($0314) komutu bir yerde JMP $EA31 gibi çalışmaktadır. O halde neden direct jump yerine indirect jump komutu kullanılmaktadır. Bana göre cevap esnek bir işletim sistemi tasarlandığı içindir. Gerçektende işletim sistemi tarafından kullanılan bu RAM vektör adresleri sayesinde C64 için çok güzel programlar yapılmıştır. Bu RAM vektörler sayesinde kullanıcı işletim sisteminin çalışmasını aksatmadan araya girip kendi rutinlerini aynı anda çalıştırabilir veya işletim sistemi rutinlerini kısmen devre dışı bırakabilir. Konumuz interruptlar olacağı için sadece interrupt (IRQ) vektörü üzerinde duracağız.

IRQ vektörünü değiştirmeden önce mutlaka SEI komutu kullanılmalıdır. Çünkü işletim sistemi zaten o anda kendi interupt rutinini çalıştırmaktadır. Siz $0314 veya $0315 adresinin içeriğini değiştirdiğiniz anda işletim sistemi bu vektöre uğruyor olabilir. İşletim sistemi artık rasgele bir adresten çalışmaya başlayacaktır ve belkide kilitlenecektir. Mesela siz $0314 adresine o an için $00 değerini yazmış olun ve işletim sistemide JMP ($0314) komutunu uyguluyor olsun. İşletim sistemi normalde $EA31 adresine uğraması gerekirken birden $EA00 adresine gidecektir. Artık ne yapar, nasıl çalışır bilinmez.

İşte bunun için IRQ vektörünü değiştirmeden önce SEI komutunu kullanıp işlemciye sen şimdilik interruptları unut, biraz bekle diyoruz. IRQ vektörünü değiştirip CLI komutuyla işlemciye kaldığın yerden devam et diyoruz. Bu arada kendi IRQ rutinimizide yazmalıyız ki CLI komutundan sonra işlemci IRQ vektörü üzerinden bizim rutinimize uğradığında çalıştırabilecek bir şeyler bulsun.

Kendi IRQ rutinimizi yazdık ama bir sorun var gibi. Normalde bir assembler programı yazıldığında, program bir kez çalışacaksa program sonunda "RTS" komutu, sürekli çalışacaksa "JMP prg. başlangıcı" komutu olmalıdır. IRQ rutinlerinde ise işiniz bittikten hemen sonra mutlaka JMP $EA31 veya JMP $EA81 komutu kullanılmalıdır. Aslında JMP $EA81 komutu $EA31 adresindeki rutinin sonunda yer alır ve ikisinin arasında kursör yakıp söndürme, klavye tarama gibi işlemler yapılır. Eğer JMP $EA31 kullanırsanız kendi IRQ rutininiz çalışırken hiç bir şey olmamış gibi klavyeyi kullanabilir veya bir başka assembler rutini çalıştırabilirsiniz. Eğer $EA81 kullanırsanız sadece IRQ rutinleri çalışmış olacaktır. Dikkat ettiyseniz -JMP $EA31 kullanırsanız hiç bir şey olmamış gibi klavyeyi kullanabilir veya başka bir assembler rutin çalıştırabilirsiniz- ifadesini kullandım. Yani aynı anda iki program çalışabilir gibi bir durum var. Şimdi biraz başa dönelim ve interrupt kelimesinin anlamlarına bir daha bakalım. "Akışını durdurmak, yarıda kesmek" demişiz. İnterrupt mikroişlemcinin içinde saniyede 60 kez oluşur, yani asıl çalışan program saniyede 60 kez durdurulur, interrupt rutini çalışır ve işlem bittikten sonra asıl program kaldığı yerden çalışmaya devam eder. Bu nedenle interrupt rutiniyle asıl program aynı anda çalışıyormuş gibi görünür. (SEI komutu işlemci içinde interrupt işlemlerini durdurur)

Özetleyecek olursak;

1- İnterrupt vektörü $0314 ve $0315 adreslerinde bulunur.
2- Bu vektörü değiştirmeden önce SEI komutu kullanılmalıdır.
3- Vektör değiştirildikten sonra CLI komutu kullanılmalıdır.
4- Eğer vektörü değiştirdiyseniz kendi rutininiz mutlaka olmalıdır.
5- Kendi rutininizin son komutu mutlaka JMP $EA31 veya JMP $EA81 olmalıdır.
6- İnterrupt saniyede 60 kez meydana geldiği için kendi interrupt rutininiz içinde sonsuz döngü veya çok sayıda iç içe döngü olmamalıdır. Yani rutininizin çalışma süresi 1/60 saniyeden uzun olmamalıdır.

İşte örnekler....

```
                *=$0900

start           sei             ;İnterruptları durdur
                lda   #<user    ;kullanıcı rutini adresinin LOW byte'ını
                sta   $0314     ;IRQ vektörü LOW byte adresine yaz
                ldx   #>user    ;kullanıcı rutini adresinin HIGH byte'ını
                sta   $0315     ;IRQ vektörü HIGH byte adresine yaz
                cli             ;İnterruptları serbest bırak
                rts             ;BASIC'e dönüş
user            jmp   $ea31     ;İşletim sistemi IRQ rutinine geri dön
                .end
```

İlk örnekte sadece irq vektörünü nasıl değiştireceğimizi öğreniyoruz. Kendi rutinimizde hiçbir işlem yapmayıp doğrudan işletim sisteminin intterrupt rutinine geri dönüyoruz.

2. örnek

```
                *=$0900

start           sei             ;İnterruptları durdur
                lda   #<user    ;kullanıcı rutini adresinin LOW byte'ını
                sta   $0314     ;IRQ vektörü LOW byte adresine yaz
                ldx   #>user    ;kullanıcı rutini adresinin HIGH byte'ını
                sta   $0315     ;IRQ vektörü HIGH byte adresine yaz
                cli             ;İnterruptları serbest bırak
                rts             ;BASIC'e dönüş
user            inc   $d020     ;Ekranın dış rengini değiştir.
                jmp   $ea31     ;İşletim sistemi IRQ rutinine geri dön.
                .end
```

2. örnekte ise çok basit bir şekilde "flashing border" efekti yaptık. Bunun için INC $D020 komutunu kullandık. $D020 adresinde ekranın BORDER dediğimiz dış kısmının renk değeri bulunur. "inc $d020" komutu ise bu adresin değerini 1 arttırır. Böylece renk değişmiş olur. Bu işlem saniyede 60 kez yapıldığı için ekran flash yapıyor gibi gözükür.

3. örnek

```
                *=$0900

start           sei             ;İnterruptları durdur
                lda   #<user    ;kullanıcı rutini adresinin LOW byte'ını
                sta   $0314     ;IRQ vektörü LOW byte adresine yaz
                ldx   #>user    ;kullanıcı rutini adresinin HIGH byte'ını
                sta   $0315     ;IRQ vektörü HIGH byte adresine yaz
                lda   #$00      ;başlangıç değeri 0
                sta   cnt       ;cnt (sayaç) adresine yaz
                cli             ;İnterruptları serbest bırak
                rts             ;BASIC'e dönüş

user            dec   cnt       ;sayacı 1 azalt
                bne   exit      ;sıfır olmadıysa "exit" yazan yere git
                inc   $d020     ;sıfır olduysa ekranın dış rengini değiştir.
exit            jmp   $ea31     ;İşletim sistemi IRQ rutinine geri dön.
cnt             .byte 0
                .end
```

3. örnekte ise interrupt içinde sayaç kullanmayı gördük. İşletim sistemi rutinimize her uğrayışında sayacın değerini 1 azaltmakta ve sayaç değeri 0 olmadıysa rutinden çıkmaktadır. Eğer sayaç değeri 0 olduysa ekranın dış rengi değiştirilmektedir. Yaklaşık 6-7 saniyede bir ekranın dış rengi değişmektedir. Eğer daha kısa zaman süresi isterseniz sayaç değerini belli değer için kontrol etmeniz gerekmektedir. Mesela sayaç değerini $40 (64) olarak belirleyin. Yaklaşık 2 saniyede bir renk değişecektir. Şu anki haliyle sayaç değeri 0 gözükse bile aslında 256 dır. Nasıl oluyor derseniz açıklayalım. Sayacın ilk değeri sıfırdır. Rutinimizin ilk komutu "DEC CNT" olduğundan sayacımız bir azalmakta ve 255 olmaktadır. Rutinimize her uğrayışta sayaç bir azaltıldığı için ancak 256. kez uğrandığında tekrar 0 olmakta ve renk değiştirme işlemi yapılmaktadır.

TARAMA SATIRI İNTERRUPTI (RASTER LINE INTERRUPT)

C64'te çeşitli interrupt kaynakları vardır. Bunlardan 4 tanesi VIC ile, 5 tanesi CIA dediğimiz 6526 ile ilgilidir. Aslında C64'te 2 adet 6526 olduğu için C64'teki interrupt kaynağı 14 adettir diyebiliriz.

VIC'e ait olan interruptlar şunlardır.

1- Tarama satırı interruptı
2- Sprite-arkaplan çarpışması interruptı
3- Sprite-sprite çarpışması interruptı
4- Light pen interruptı

CIA'lara ait olan interruptlar ise şunlardır.

1- A zamanlayıcısı interruptı
2- B zamanlayıcısı interruptı
3- Gün saati alarm interruptı
4- Seri port dolu/boş sinyali interruptı
5- /FLAG (user port'ta bulunur) girişi interruptı

Yukarıdaki interrupt kaynaklarından en çok kullanılanı tarama satırı interruptıdır. Tarama satırı (raster) bir ekran görüntüsü oluşturulurken VIC tarafından üretilir ve bu satırın (rasterin) değeri $D012 adresinde tutulur ve sürekli değişir. $D012 adresi çift fonksiyonlu bir register olup okunduğunda o anki raster konumunun en alt 8 bitidir. 9.bit ise $D011 adresindeki registerin en soldaki (7.bit) bitidir. Normalde kullanılabilen ekran bölümü 51 ($33) ile 251 ($FB) raster satırları arasındadır. $D012'ye ve $D011'in 7. bitine yazılacak bir değer iç raster karşılaştırmasında kullanılır. O anki raster değeri

ile yazılmış olan değer birbirine eşitse raster interrupt latch biti "1" olur.

Kafanızın karışmaya başladığından eminim. Ama güzel bir intro veya demo yapabilmek için bunları ve daha fazlasını bilmeniz gerekmektedir.

Raster interruptını kullanmadan önce yapılacak bir işlem daha vardır. $D01A adresindeki interrupt enable registerine hangi interruptı kullanacağımızı bildirmemiz gerekmektedir. $D01A adresine $1 sayısını yazdığımızda artık raster interruptını kullanabiliriz. Elimizde bir adres daha var $D019 adresi. $D019'da bir interrupt oluştuğunda $D01A ile seçilen interrupta ait bit "1" olur ve programcı hangi interruptın meydana geldiğini anlayabilir. Fakat $D019'daki ilgili bit "1" olduktan sonra hep öyle kalır. Bir sonraki interrupt için bu adresin temizlenmesi gerekir. Şimdi yazdıklarımızı toparlayalım ve basitleştirelim.

1- $D01A adresine $1 yazarak raster interruptını kullanacağımızı belirleriz.
2- $D012 adresine hangi rasterde interrupt yapacağımızı yazarız.
3- $D019 adresini interrupt rutini içinde temizleriz.
4- $D012'yi interrupt rutini içinde kontrol edip rutinimizi çalıştırırız.

Bütün bu öğrendiklerimizle ilgili bir örnek verelim.

```
                *=$0801
                .word nextline
                .word 2004          ; 2004
                .byte $9e           ; SYS
                .text "2061"        ; 2061
                .byte 0             ; komutu
nextline        .word 0

start           sei                 ;interruptları durdur
                lda   #<irq         ;interrupt vektörünü
                ldx   #>irq         ;kendi rutinimizi
                sta   $0314         ;gösterecek şekilde
                stx   $0315         ;değiştir.
                lda   #$30          ;raster değeri
                sta   $d012         ;raster registerine
                lda   #$01          ;raster interrupt yapacağımızı
                sta   $d01a         ;interrupt enable registerine bildir.
                cli                 ;interruptları serbest bırak.
                Jmp   *             ;sonsuz döngüye gir.

irq             inc   $d019         ;bir sonraki interrupt için temizle
                lda   #$30          ;bu değeri
wait0           cmp   $d012         ;raster registeri ile karşılaştır.
                bne   wait0         ;eşit değilse bekle. Eşit ise devam et
                lda   #$01          ;aküye 1 sayısı
                ldx   #$0b          ;X'e $0b
                ldy   #$0a          ;Y'ye $0a
tekrar1         dex                 ;x registerini 1 azalt
                bne   tekrar1       ;0 oluncaya kadar tekrarla
                sta   $d020         ;0 olduysa aküdeki değer ekranın dış rengi ve
                sta   $d021         ;ekranın iç rengi olsun
tekrar2         dey                 ;y registerini 1 azalt
                bne   tekrar2       ;0 oluncaya kadar bekle. 0 olduysa
                sty   $d020         ;ekranın dış rengi ve
                sty   $d021         ;ekranın iç rengi olsun
                jmp   $ea81         ;işletim sistemine geri dön.

                .end
```

X registerini renk şeritinin başlangıcını sabitlemek için, Y registerini ise bitişini sabitlemek için kullanıyoruz. Daha doğrusu çizginin zamanlaması için kullanıyoruz. Eğer bu sayıları değiştirirseniz çizginiz titremeye başlar.

PROGRAM DÖKÜMLERİ

PROGRAM ADI: KARSET5 0801 0860

```
0801: 0B 08 D4 07 9E 32 30 36  - 43A8
0809: 31 00 00 00 78 A9 33 85  - 5108
0811: 01 A9 D0 A2 30 A0 00 84  - 6D02
0819: FB 85 FC 84 FD 86 FE A2  - C027
0821: 10 B1 FB 91 FD C8 D0 F9  - CB6B
0829: E6 FC E6 FE CA D0 F2 A9  - D7A5
0831: 37 85 01 58 A9 1C 8D 18  - 4C35
0839: D0 A9 30 A0 00 84 FB 85  - 8AB9
0841: FC A2 10 86 FD C8 C8 C8  - B70B
0849: A2 04 B1 FB 4A 11 FB 91  - 8C6D
0851: FB C8 CA D0 F5 C8 D0 ED  - DBBB
0859: E6 FC C6 FE D0 E7 60 ED  - D11C
```

PROGRAM ADI: KARSET6 0801 0869

```
0801: 0B 08 D4 07 9E 32 30 36  - 43A8
0809: 31 00 00 00 78 A9 33 85  - 5108
0811: 01 A9 D0 A2 30 A0 00 84  - 6D02
0819: FB 85 FC 84 FD 86 FE A2  - C027
0821: 10 B1 FB 91 FD C8 D0 F9  - CB6B
0829: E6 FC E6 FE CA D0 F2 A9  - D7A5
0831: 37 85 01 58 A9 1C 8D 18  - 4C35
0839: D0 A9 30 A0 00 84 FB 85  - 8AB9
0841: FC A2 10 B1 FB 29 F0 48  - 8D99
0849: 0A 85 FD 68 05 FD 85 FE  - A20F
0851: B1 FB 29 0F 48 4A 85 FD  - 86CA
0859: 68 05 FD 05 FE 91 FB C8  - A895
0861: D0 E1 E6 FC CA D0 DC 60  - BF9B
```

PROGRAM ADI: BALLS 0801 0938

```
0801: 0B 08 D4 07 9E 32 30 36  - 43A8
0809: 31 00 00 00 78 20 44 E5  - 5614
0811: CA 8E 21 D0 A9 23 9D F8  - 9E4E
0819: 07 BD 00 09 9D 00 D0 BD  - 6F65
0821: 08 09 9D 08 D0 A9 0F 9D  - 6869
0829: 27 D0 E8 E0 08 D0 E5 A9  - AAE7
0831: FF 8D 15 D0 8D 1C D0 A2  - 909E
0839: 0B A0 0C 8E 25 D0 8C 26  - 5F7E
0841: D0 A9 5A A2 08 8D 14 03  - 4E09
0849: 8E 15 03 A2 01 8E 1A D0  - 665B
0851: A0 80 8C 10 D0 58 4C 57  - 68A5
0859: 08 EE 19 D0 AD 12 D0 D0  - 94CE
0861: FB AD 91 08 D0 23 A9 0A  - 670F
0869: 8D 91 08 AE 90 08 E0 20  - 6666
0871: D0 05 A2 08 8E 90 08 A0  - 69CF
0879: 00 BD 10 09 99 01 D0 E8  - 79F0
0881: C8 C8 C0 10 D0 F3 EE 90  - B2CD
0889: 08 CE 91 08 4C BC FE 00  - 6ACF
0891: 00 00 00 00 00 00 00 00  - 0000
0899: 00 00 00 00 00 00 00 00  - 0000
08A1: 00 00 00 00 00 00 00 00  - 0000
08A9: 00 00 00 00 00 00 00 00  - 0000
08B1: 00 00 00 00 00 00 00 00  - 0000
08B9: 00 00 00 00 00 00 00 00  - 0000
08C1: 55 00 01 77 40 05 DD D0  - 6F05
08C9: 07 7F B0 15 EE E4 17 FB  - 9763
08D1: BC 1D EE AC 57 FA A9 5D  - 96AC
08D9: EA AB 77 BA AB 5F EA AB  - AB51
08E1: 77 BA AB 5D EA AB 57 FA  - AC35
08E9: A9 15 EA A8 17 FA AC 15  - 7C24
08F1: EE E4 07 7B B0 05 FF D0  - 9E1A
08F9: 01 77 40 00 55 00 00 3C  - 26C5
0901: B0 5C B0 7C B0 9C B0 BC  - A3C0
0909: B0 DC B0 FC B0 1C B0 B0  - A67C
0911: B0 B0 B0 B0 B0 B0 B0 B8  - B1D8
0919: C0 C8 D0 D8 E0 D8 D0 C8  - D080
0921: C0 B8 B0 B8 C0 C8 D0 D8  - C650
0929: E0 D8 D0 C8 C0 B8 B0 B8  - C0B0
0931: C0 C8 D0 D8 E0 D8 D0 B8  - CCD0
```

PROGRAM ADI: HBASSON 0801 0BFA

```
0801: 0B 08 D4 07 9E 32 30 36  - 43A8
0809: 31 00 00 00 A9 32 A2 08  - 3ABC
0811: 85 FB 86 FC A9 00 A2 C0  - 9EB5
0819: 85 FD 86 FE A2 04 A0 00  - 723E
0821: B1 FB 91 FD C8 D0 F9 E6  - DB3D
0829: FC E6 FE CA D0 F2 4C 00  - 97D8
0831: C0 A9 3D A2 08 8D 08 03  - 4872
0839: 8E 09 03 60 A5 7A A6 7B  - 71CA
0841: 85 FB 86 FC A9 52 A2 0B  - 80D8
0849: 85 FD 86 FE A2 00 A0 00  - 71AA
0851: 20 73 00 85 02 B1 FD C5  - 8979
0859: 02 D0 13 C8 98 DD B4 0B  - 79A5
0861: D0 EE 8A 0A AA AD 2B 0B  - 6405
0869: 48 BD 2A 0B 48 60 BD B4  - 762D
0871: 0B 18 65 FD 85 FD 90 02  - 7471
0879: E6 FE E8 EC B3 0B D0 D5  - BB5F
0881: A5 FB A6 FC 85 7A 86 7B  - 9C98
0889: 4C E4 A7 20 73 00 F0 12  - 611C
0891: 20 9E B7 E0 01 90 11 E0  - 82F3
0899: 1B B0 0D CA 20 FF E9 4C  - 8678
08A1: AE A7 20 44 E5 4C AE A7  - 8B99
08A9: 4C 48 B2 20 9B B7 8E 20  - 6958
08B1: D0 4C AE A7 20 9B B7 8E  - 8E51
08B9: 86 02 4C AE A7 20 9B B7  - 7F6F
08C1: 8E 21 D0 4C AE A7 20 C3  - 8673
08C9: 09 B0 06 2D 15 D0 4C D5  - 712A
08D1: 08 0D 15 D0 8D 15 D0 4C  - 61E2
08D9: AE A7 20 E0 09 20 FD AE  - 8A89
08E1: 20 EB B7 8E E6 0B AE E7  - A2A4
08E9: 0B A5 15 F0 09 BD E8 0B  - 6E26
08F1: 0D 10 D0 4C FD 08 BD F0  - 92B9
08F9: 0B 2D 10 D0 8D 10 D0 8A  - 7193
0901: 0A AA A5 14 9D 00 D0 AD  - 7A91
0909: E6 0B 9D 01 D0 4C AE A7  - 84D6
0911: 20 E0 09 20 EB 09 8A AE  - 7325
0919: E7 0B 9D F8 07 4C AE A7  - 888B
0921: 20 C3 09 B0 06 2D 1D D0  - 6128
0929: 4C 2F 09 0D 1D D0 8D 1D  - 49A8
0931: D0 20 EB 09 20 C9 09 B0  - 7140
0939: 06 2D 17 D0 4C 43 09 0D  - 34B1
0941: 17 D0 8D 17 D0 4C AE A7  - 86C4
0949: 20 C3 09 B0 06 2D 1C D0  - 60FF
0951: 4C 57 09 0D 1C D0 8D 1C  - 4C46
0959: D0 4C AE A7 20 9B B7 8E  - 8E51
0961: 25 D0 20 EB 09 8E 26 D0  - 7B07
0969: 4C AE A7 20 E0 09 20 EB  - 7C4D
0971: 09 8A AE E7 0B 9D 27 D0  - 82D9
0979: 4C AE A7 20 C3 09 B0 06  - 5B13
0981: 2D 1B D0 4C 8A 09 0D 1B  - 3A9B
0989: D0 8D 1B D0 4C AE A7 20  - 76BD
0991: 9B B7 20 C9 09 B0 03 A9  - 737A
0999: 00 2C A9 FF 4C D5 08 20  - 5FFB
09A1: 9B B7 E0 19 B0 19 8E F8  - 97A8
09A9: 0B 20 EB 09 E0 28 B0 0F  - 59CA
09B1: 8E F9 0B AE F8 0B AC F9  - A52A
09B9: 0B 20 0C E5 4C AE A7 4C  - 6CBD
09C1: 48 B2 20 E0 09 20 EB 09  - 5BAB
09C9: E0 02 B0 F3 E0 01 F0 07  - 7DAF
09D1: AE E7 0B BD F0 0B 60 AE  - 893A
09D9: E7 0B BD E8 0B 38 60 20  - 5B98
09E1: 9B B7 E0 08 B0 D9 8E E7  - AD90
09E9: 0B 60 20 FD AE 20 9E B7  - 8491
09F1: 60 20 73 00 D0 03 4C 08  - 3AB8
09F9: AF C9 56 F0 6E C9 48 D0  - A34B
0A01: F5 A9 00 A2 18 8D E1 0B  - 6D05
0A09: 8D E5 0B 8E E2 0B AC E5  - 98B7
0A11: 0B AE E1 0B 20 5A 0A A5  - 5B9E
0A19: FB 85 FD A5 FC 85 FE A5  - C468
0A21: 02 85 04 A5 03 85 05 B1  - 5A3C
0A29: FD 48 B1 04 48 AE E2 0B  - 6F19
0A31: 20 5A 0A B1 02 91 04 B1  - 5C63
0A39: FB 91 FD 68 91 02 68 91  - 81DD
0A41: FB C8 C0 28 D0 CB A0 00  - 8572
0A49: 8C E5 0B EE E1 0B CA 8E  - 9428
0A51: E2 0B E0 0C D0 BB 4C E4  - 9A3E
0A59: A7 BD F0 EC 85 FB 85 02  - 938F
0A61: BD C8 0B 85 FC 18 69 D4  - 8F1C
0A69: 85 03 60 A2 00 A9 04 86  - 5BFB
0A71: FB 85 FC 86 FD 18 69 D4  - A424
0A79: 85 FE A2 00 A9 27 8D E4  - 8F78
0A81: 0B A0 00 8C E3 0B AC E4  - 89A3
0A89: 0B B1 FB 48 B1 FD 48 AC  - 99BF
0A91: E3 0B B1 FD AC E4 0B 91  - 956A
0A99: FD AC E3 0B B1 FB AC E4  - BD03
0AA1: 0B 91 FB AC E3 0B 68 91  - 84AE
0AA9: FD 68 91 FB CE E4 0B C8  - ABBC
0AB1: C0 14 D0 CF A5 FB 18 69  - 8CAC
0AB9: 28 85 FB 85 FD 90 04 E6  - 9B3E
0AC1: FC E6 FE E8 E0 19 D0 B4  - BC71
0AC9: 4C E4 A7 A9 12 20 D2 FF  - 9B53
0AD1: 4C E4 A7 20 73 00 F0 0D  - 5FF5
0AD9: 20 9E B7 E0 08 B0 0E 8A  - 741F
0AE1: 0A 18 69 10 2C A9 14 8D  - 4F25
0AE9: 18 D0 4C AE A7 4C 48 B2  - 7F55
0AF1: 20 73 00 F0 04 20 9E B7  - 6E70
0AF9: 2C A2 33 8E 05 DC 4C AE  - 784A
0B01: A7 20 9B B7 E0 02 B0 12  - 6BDF
0B09: AD 11 D0 E0 01 F0 03 29  - 658D
0B11: EF 2C 09 10 8D 11 D0 4C  - 5C1E
0B19: AE A7 4C 48 B2 20 9B B7  - 841D
0B21: 20 B3 EE CA D0 FA 4C AE  - AD45
0B29: A7 8B 08 AB 08 B4 08 BD  - 7152
0B31: 08 C6 08 DA 08 10 09 20  - 34BD
0B39: 09 48 09 5C 09 6B 09 7B  - 3F84
0B41: 09 8F 09 9F 09 F1 09 CB  - 722E
0B49: 0A D3 0A F0 0A 01 0B 1D  - 3636
0B51: 0B 43 4C 53 42 B0 44 45  - 5228
0B59: 52 49 4E 4B 50 41 50 45  - 4A2A
0B61: 52 53 50 52 49 54 45 53  - 4F3E
0B69: 50 52 B9 53 50 52 50 4E  - 59C0
0B71: 54 53 50 52 BD 53 50 52  - 5ECF
0B79: 4D 55 4C 53 50 52 4D 43  - 4CF7
0B81: 4C 53 50 52 43 4F 4C 53  - 4E84
0B89: 50 52 49 B0 53 50 52 41  - 579F
0B91: 4C 4C 43 55 52 53 B0 4D  - 5E32
0B99: 49 52 52 B0 49 4E 56 46  - 5842
0BA1: 91 54 53 50 45 45 44 53  - 50BB
0BA9: 43 52 45 45 4E 44 45 4C  - 4864
0BB1: 41 59 14 03 05 03 05 06  - 1048
0BB9: 04 06 04 06 06 06 05 06  - 0587
0BC1: 05 05 03 03 05 06 05 04  - 0482
0BC9: 04 04 04 04 04 04 05 05  - 0464
0BD1: 05 05 05 05 06 06 06 06  - 05A8
0BD9: 06 06 06 07 07 07 07 07  - 06C5
0BE1: 00 00 00 00 00 00 00 01  - 003B
0BE9: 02 04 08 10 20 40 80 FE  - 5F20
0BF1: FD FB F7 EF DF BF 7F 00  - 9FA5
0BF9: 00 00 00 00 00 00 00 00  - 0000
```

Geçen sayıda yazdığım gibi HADES BASIC'in tam listesi. Örnekler ise maalesef yetişmedi. Bir sonraki sayıda görüşebilmek dileğiyle hoşça kalın.