C64 TÜRKİYE

SAYI : #$05 **TEMMUZ 2004**

BİLGİ PAYLAŞTIKÇA ARTAR

DEMO EFEKTLERİ #1 (3.BOYUT)
HADES BASIC #2
PROGRAM BÖLÜMÜ
ROM RUTİNLERİ #2
OYUN MERKEZİ
PROGRAM DÖKÜMLERİ
SON SAYFA

C64 TÜRKİYE

SAYI : #$05 TEMMUZ 2004
BİLGİ PAYLAŞTIKÇA ARTAR

İÇİNDEKİLER

İÇİNDEKİLER

1- *Dergi için COMMODORE 64 ile ilgili her türlü yazı, resim, program, oyun tanıtımı, ipuçları, hile ve ufak-tefek rutinler gibi malzemeler kabul edilir.*
2- *Yukarıdaki malzemelerin çalışır durumda olduğundan emin olunmalıdır.*
3- *Gönderilen malzemeler dergide mutlaka yayınlanacak diye herhangi bir şart yoktur.*
4- *Malzemeyi gönderenin adı-soyadı-e-mail adresi-oturduğu şehir ve kendisini tanıtan kısa bir not yazması tercih edilir.*
5- *Derginin editörü gönderilen malzemelerde değişiklik, düzenleme vs yapabilir.*
6- *Gönderilen malzemelerin sorumluluğu gönderene aittir.*
7- *Hiç bir şekilde siyasi, dini yazılara yer verilmeyecektir. Ayrıca kişilere ve şirketlere yönelik hakaret, aşağılama, küfür gibi yazılar yayınlanmayacaktır.*
8- *Emeğe saygı gereği kaynak göstermek şartıyla herkes dergideki yazılardan alıntı yapabilir.*
9- *BU DERGİ KESİNLİKLE TİCARİ AMAÇLI DEĞİLDİR VE PARA İLE SATILAMAZ.*
10- *Dergiyi okuyan herkes bütün bu maddeleri kabul etmiş sayılır.*

C64 TÜRKİYE
Hazırlayan ve sahibi : İSMAİL **"HADES"** ŞAHİN
Katkıda bulunanlar :
BİLGEM **"NIGHTLORD"** ÇAKIR
nightlord@nightlord.dr2.net
YEŞİM **"INSOMNIA"** GÜLEN
insomnia@hi3s.org

EDİTÖRÜN KLAVYESİ

Uzun bir aradan sonra herkese merhaba.....
Aslında C64 Türkiye dergisini çıkarmaktan vazgeçmiştim. Fakat düşündümki eğer vazgeçersem ülkemizde C64'ü herhangi bir şekilde kullanmaya başlamış olanları yalnız başlarına bırakmış olacaktım. İşte bu nedenle kaldığımız yerden devam edelim dedim. Bu arada dergide bir değişiklik oldu. Umarım son olur. Quark Xpress ile hazırlamaya başladığımız dergi, Deniz arkadaşımızın artık dergi ile uğraşmayacak olmasından dolayı, bu sayıdan itibaren tamamen word ile hazırlanmış olacak ve PDF'e çevrilecek.

İyi kötü 5. sayıya geldik ve gördüğünüz gibi tamamen programlama ağırlıklı. Haber köşesi azaldı. Şimdide kısa açıklamalar yapalım.
İlk konumuz C64 ile 3 boyutlu efektler hakkında bir giriş yazısı. Yazıyı hazırlayıp gönderen BİLGEM "NIGHTLORD" ÇAKIR arkadaşımıza teşekkür eder ve çalışmalarının devamını bekleriz.
HADES' BASIC ile C64'ü geliştirmeye devam ediyoruz. Bu sayıdaki yeni komutlarımız sprite'larla ilgili ve önümüzdeki sayıda da yeni komutlar eklemeye devam edeceğiz.
Programlama bölümünde iki kısa programımız var. Birincisi C64'te çizgi şeklinde bir kursör kullanmamızı sağlayan kısa bir programdır. İkincisi ise karakter setini baş aşağı çeviren programdır.
Rom rutinlerinde de işinize yarayacak adresleri vermeye devam ediyoruz.
Oyun merkezi yeni bir bölüm olup hafızlardan silinmeyen C64 oyunlarını tanıtacağız. Bu arada yazıyı hazırlayan YEŞİM arkadaşımıza teşekkür etmeyi unutmayalım.
Derginin daha sık çıkması yapılacak katkılara bağlıdır. Elimizde ne kadar çok materyal olursa derginin hazırlanıp çıkması daha kısa sürede olur. Özellikle haberler bölümünde desteğe ihtiyacım var. Yurt içi ve yurt dışı scene haberleri (parti haberleri, yeni çıkan demolar, oyunlar, pc üzerinde çalışan c64 ile ilgili programlar vs..) konusunda yardım etmek isteyenler bana mail atabilirler.

C64 TÜRKİYE DERGİSİ RESMİ WEB SİTESİ
www.c64turkiye.com
iletişim : c64turkiye@yahoo.com
veya hades@hi3s.org

HABERLER

- **"HI3S" GRUBU KURULDU :** Eylül 2003'te kurulan C64 scene grubu "INDEPENDENT", kişisel problemler dolayısıyla, ortak alınan bir karar sonucu dağıldı. Dağılmayı izleyen günlerde eski "independent" üyelerinden bazıları bir araya gelerek **"HI3S"** isimli yeni bir c64 scene grubu kurdu. Grubun resmi web sitesi www.hi3s.org olup grubun adı grup üyelerinin nicklerinin ilk harflerinden meydana gelmiştir. Şu anki üyeler : **HADES, INSOMNIA, SLOWHAND, SHADOW, SPUNKY.** Dağılma sonucu geçen sayıda haberini verdiğimiz C64 forumu (independentonline.org/forum) kapandı ve HI3S üyeleri tarafından açılan www.hi3s.org/forum faaliyete başladı.
- **HI3S** üyelerinden SLOWHAND, **"ONE MORE CHANCE"** isimli parçasıyla katılmış olduğu **"SID WINE music compo"** da 3. olmuştur. Başarılarının devamını dileriz.
- **7D4 DEMO PARTY YAPILDI :** 8-9 MAYIS 2004 tarihinde İKİTELLİ/İSTANBUL'da yapılan demo party'de birbirinden güzel productlar release edildi. C64 – AMIGA – PC kategorilerinin dışında birde X-BOX demosu vardı. 7D4 demo party ile ilgili herşeyi partinin resmi sitesi olan www.7dx.org adresinden öğrenebilirsiniz.
- **HI3S** grubu olarak partiye "little secret" isimli parçayla music compo'ya, "magic scroll" ve "8x8 big scroll" isimli ürünlerlede 256 byte demo compo'ya katıldık. Bunların dışında ilk intromuz ve first release olarak **"BLUE ILLUSIONS"** zak collection release edildi.

İlk intro

"Blue Illusions" Zak Collection

Çalışmalarımızı www.hi3s.org adresinden indirebilirsiniz.

- 7D4 demo party'nin düzenlenmesinden hemen sonra Türkiye'nin en eski C64 ve AMIGA gruplarından olan BRONX'ta bazı gelişmeler oldu. Elimizde kesin bir bilgi olmamakla beraber grupta bölünmelerin olacağı ve yeni oluşumların ortaya çıkacağı tahmin ediliyor.
- Almanca olarak yayınlanan LOTEK64 dergisinin 10. sayısı çıktı.

DEMO EFEKTLERİ

ÜÇÜNCÜ BOYUT

HAZIRLAYAN : **BİLGEM ÇAKIR**
İLETİŞİM : **nightlord@nightlord.dr2.net**
NEREDEN : **İZMİR**
NİCK / GRUP(LAR) : **NIGHTLORD / AESRUDE / CIVITAS**
WEB SİTE : **http://nightlord.dr2.net**

Herkese merhaba... Bu bölümde sizlere 3D demo efektleri dünyasını tanıtmaya çalışacağım. Yazıları takip ettikçe bu tür efektlerin arkasında yatan temel matematik ve geometrik prensipleri öğrenecek, ve ardından bunları C64'de assemblerda nasıl gerçekleştirebileceğinizi göreceksiniz. Bu sırada yer yer scene programcılığı, yazılım mühendisliği, bilgisayar mimarisi gibi çeşitli konulara da değinmeyi planlıyorum. Ama önce biraz felsefe. "Eee hani matematik öğrenecektik" demeyin. Felsefe bütün bilimlerin babasıdır. Düşünüyoruz öyleyse varız hesabı.

Neden Demolar, Neden C64 ve Neden 3D?

Herhangi bir konuda oturup emek harcamanız gerektiğinde, bu emeğin karşılığında ne kazanacağınızı, ne öğreneceğinizi sorgulamak hem hakkınız, hem de yol boyunca ihtiyacınız olacak olan motivasyonun kaynağı olacak cevaplara giden yolunuzdur. Öyle ya, neden şimdi durduk yerde 20 yıllık eski bir bilgisayarda, günümüzde ucuz ve hızlı donanım sistemlerinin kolaylıkla yapabildiği bazı şeyleri zorlaya zorlaya yapmaya çalışasınız ki? Bu büyük soruyu kendi kendinize cevaplayıp bu işlere girişmek konusunda kendinizi dürüst bir şekilde sorgulamanız ve cevaplarınıza ikna olmanız kesinlikle çok önemli. Bu konuda bazı noktalara en baştan açıklık getirmek ve yanlış sebeplerle zamanınızı ve emeğinizi harcamanızı engellemek istiyorum:

1- Kolay olmayacak ve sizin çabanız gerekecek. Ben bu yazılarda mümkün olduğu kadar alt seviyeden başlayacağım. Ama ne olursa olsun sizin bol miktarda kafa yormanız anlamadığınız yerleri tespit edip üzerinde çalışmanız ve çözüm bulamadığınız noktalarda bana bir şekilde sorularınızı iletmeniz gerekecek. Yani bu yazıları roman okur gibi okuyup tamamlarsanız çok gelişme kaydedemezsiniz.

2- Çok zor da olmayacak. Kimi insanlar matematik kelimesini duydukları anda korkuya kapılırlar. Matematik, geometri ve bilgisayar bilimleri sistematik, tutarlı ve öğrenilebilir konulardır. Bu alanlarda kafa yormak hem zevklidir hem de size tutarlı açık ve analitik düşünmeyi öğretir.

3- Parasal getiri beklemeyin. C64 üzerindeki demo çalışmalarınızdan dolayı kimse size para ödemeyecek. Siz genel olarak çeşitli meslekleri icra ederken işinize yarayabilecek pek çok şey öğreneceksiniz elbette ama oyun veya demo yapıp satmak ve para kazanmak gibi planlarınız varsa yanlış yerdesiniz.

4- Yüksek seviyeli diller (C++/Java vs), Script dilleri (Perl, Tcl/Tk vs), Web Teknolojileri(html, .NET vs) gibi konularla uzaktan yakından ilgisi olmayan işler yapacaksınız. Eğer bu alanlarda kendinizi geliştirmek istiyorsanız yanlış yerdesiniz.

Bu noktalardan sonra gelelim ilk soruya... Neden demo programcısı olmak isteyesiniz? En basit ve dürüst cevap " çok zevkli olduğu için"... Programcılık zaten zevkli bir iştir. Bilmece çözmek, kafayı zorlamak neden zevkli ise programcılık da o yüzden zevklidir. Başlangıçta kafanızda soyut bir fikir olarak başlayan bir programın, sonunda önünüzdeki bilgisayarda çalışmasını izlemek başlı başına çok zevklidir.

Üstelik demo programlamayı başka yazılımları programlamaktan daha zevkli kılan şeyler de vardır. Demolar görsel ve işitsel yönden hoş (janjanlı) programlardır. Sadece bu yüzden bile bir fabrikada üretim hattını kontrol eden yazılımı yazmaktan çok daha zevklidir demo yazmak. Ayrıca demoların en önemli özelliği donanımdan maksimum verim almak üzerine planlanmalarıdır. Diğer yazılımlar üzerlerinde çalıştıkları donanımdan maksimum yararlanmaya çalışmaz. Hatta tam tersine mümkün olduğunca çeşitli donanımlar üstünde çalışabilecek şekilde tasarlanırlar ki daha çok müşterileri olsun.

Oysa demo programcılığının temelini oluşturan prensiplerden biri deyim yerindeyse **"donanıma takla attırmaktır"**. İşte bu takla attırma hissi de bir demo programcısına kendini en iyi hissettiren unsurlardan biridir.

Peki neden C64? C64 genel olarak iyi tasarlanmış ve karmaşıklığı ile yapabildiği işler çok "tadında" bir bilgisayardır. Kullandığı assembler dili kolay öğrenilebilir ve demo programcılığına kolayca başlanabilir bir platformdur. Ayrıca nasıl olduğunu açıklayamadığım ama kesinlikle pek çok C64 kullanıcısında gözlemlediğim üzere bir şekilde kullanıcılarında bir bağlılık ve kullanırken mutluluk üreten bir alettir. Bu yüzden insanlar Commodore firmasının batmasının ardından 10 yıl sonra bile zevkle bu bilgisayar için kafa patlatıp acayip efektler üretmekte, dergiler çıkartmakta ve hatta üç boyutlu efektler hakkında tanıtım yazıları yazmaktadır... :)

Bu noktada PC demolarına da değinelim. PC Scene'de aktif olan arkadaşların tepkisini çekmek gibi bir niyetim olmadığını fakat özellikle yeni başlayanlar için C64'ün neden daha iyi bir scene platformu olduğuna inandığımı açıklamak istiyorum. PC üzerinde de bugün demolar programlanmaktadır. Hatta C64 le uğraşanlardan çok daha fazlası PC demoları üretmektedir. Bu noktada "Demek demo programlamak istiyorum gideyim 2.4 Ghz Pentium 4 üzerinde programlayayım" diyebilirsiniz. İşte demo programcılığında platform seçimi sizin önceliklerinize bağlıdır. Kimi insanlar demolarındaki ses müzik ve sanatsal anlatıma en çok önemi verirler. Onların için demo hazırlamak, kısa film çekmek ya da kısa bi animasyon film hazırlamak gibidir. Teknik anlamda yapılması zor şeyler yapmak ikinci plandadır. Bu tip demolarda en önemli şey senaryo ve akıştır. Bu tip demolar özellikle son yıllarda bütün platformlarda artarak yapılmaktadır. PC özellikle bu tip demolar (bunlara konsept demoları da denir) yapmak isteyenler için kesinlikle en uygun platformdur. Bu arada C64'te de son yıllarda konsept demolarında artış olduğu söylenebilir.

Kimi insanlarsa demolarda teknik açıdan şov yapmaya çalışır. İzleyenlere imkansızmış gibi gözüken efektler üzerinde uğraşırlar. Özellikle ya daha önce yapılmamış efektler ya da daha önceki yapılan örneklerini hız veya başka özellikler bakımından yenen efektler yapmak bu yaklaşımdakilerin amacıdır. Teknik anlamda bu tip efektler üzerinde çalışıp başarılı olabilmek için mutlaka ve mutlaka makine dili kullanılır. Ayrıca bir işletim sistemi üzerinde bu tip efektleri yapmak da kendi içinde yavaşlamalara sebep olur. Bu bakımlardan C64 teknik anlamda kendini ve donanımı çok zorlamak isteyen insanlar için gerçek bir hacker makinesidir. Bazı efekt dallarında (özellikle matematiksel efektlerde) gerçekten istediğiniz şeyi C64'te kesinlikle yapamayacağınızı düşünüyorsanız (ki böyle efektler olabilir. Ama geçmişte imkansız olduğu söylenen texture mapping, doom türü efektlerin bir noktada gerçek olduğunu unutmamalı) o zaman PC gerçekten sizin için doğru seçim olabilir. O halde bile C64 üzerinde geçirilmiş bir iki yıldan sonra PC'ye geçmek, direkt PC'de başlamaktan çok daha güçlü kılabilir sizi.

Bu son iki paragrafta anlatılan demo yaklaşımlarına sırasıyla kavramsal ve teknik yaklaşımlar diyebiliriz. Bu söylediklerimden PC'de hiç teknik demo yapılmadığı gibi bir anlam çıkarılmamalı. Ancak PC'de teknik demo programcılığı yapmak birazdan anlatacağım üzere daha zor 'ölçülebilir' sonuçlar üretir.

Demo programcıları arasındaki rekabet açısından da C64 bazı avantaj ve dezavantajlar sağlamaktadır. En önemli avantaj C64ün donanımının değişmemesidir. CPU grafik ve ses çipleri hep aynıdır. Yani bütün programcılar tamı tamına eşit kaynaklarla yola çıkar ve kim daha akıllıca algoritmalar kullanır ve trikler uygularsa o kazanır. Çünkü demoları dünyada izleyen herkes aynı hızda ve etkinlikte izler. Oysa PC platformunun sabit güçte olmaması yüzünden bir demoyu GeForce4 ile izleyen birisi çok beğenirken TNT ile izleyen birisi çok kötü bulabilir. Şahsen ben PC demolarını izlerken gördüğüm şeyin ne kadarının programcının başarısı ne kadarının donanımın başarısı olduğu hakkında hep şüpheye düşmüşümdür. Bana Formüla 1 yarışlarını çağrıştırır.

Herkes farklı teknolojide arabalarla yarışırken birinci olan, arabası yüzünden mi pilotu yüzünden mi oluyor? Tabii orada arabayı yapanlarla süren kişi bir takım ve başarı takımın başarısı. Ama demo programcıları ile grafik kartı tasarımcıları beraber çalışmıyor.

C64ün bu konudaki dezavantajı ise dünyada pek çok sayıda süper iyi C64 programcısının bulunmasıdır. 20 yıllık başarılı bir geçmişin birikimi bugün yeni başlayan bir programcı için hemen rekabete girilemeyecek kadar güçlüdür. Bugün scene de aktif olan ve belki hayatınız boyunca çalışsanız geçemeyeceğiniz 5-6 adam olabilir. Ama denemeye değer. Bu konuda genelde iş hayatına yeni başlayan genç mühendislere söylediğim bir şeyi paylaşmak istiyorum. Hiç kimseyi gözünüzde büyütmeyin ve hiç kimseyi hafife almayın.

Peki neden 3D? Eğer C64 demo programcısı olmak konusunda ikna olmuş iseniz üzerinde çalışabileceğiniz iki temel yaklaşım vardır. Birincisi C64ün donanımının çeşitli özelliklerini kullanarak yapılan özellikle de grafik çipine resmen takla attıran efektler üzerine gitmek. Bir diğeri de diğer platformlarda da yapılan bazı efektleri o platformlardan çok daha güçsüz olan C64 üzerinde yapmaktır. Bu efektler genelde sıkı matematik bilgisi üzerine kurulu efektlerdir. İşte 3D efektleri bu sınıfa girer. Eğer iyi bir demo programcısı olmak istiyorsanız aslında bu iki grup efekt üzerinde de yavaş yavaş kendinizi geliştirmelisiniz. Yalnızca bir gruptaki efektleri yaparak listelerde gelebileceğiniz nokta genelde ilk 30un içine girmektir. İlk 10a girmenin yolu iki grup efektten de yapabilmektir.

Hala neden 3D sorusunu cevaplamadım. Çünkü bir cevabı yok. Nasıl olsa bu efektleri öğrenmeye bir yerden başlamanız lazım. Örneğin ben bu grup efekti yapmayı daha zevkli bulacağımı düşündüğüm için bunlara girişmiştim. Siz de aslında ne tür efektlerden hoşlanıyorsanız onlar üstünde çalışmalısınız. Sonuçta unutmayın en önemli şey bu işten zevk almanız.

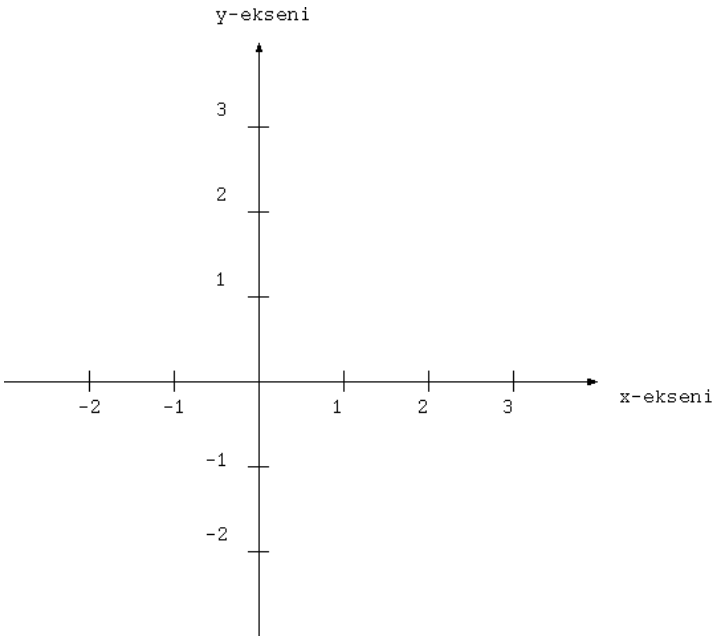
Bu kadar felsefeden sonra artık teknik tarafa geçelim. Şu anda demo programcısı olmaya karar vermiş, C64ün sizin için doğru platform olduğuna ikna olmuş ve 3D efektlerini bizzat yapmadan rahat edemeyeceğinizi anlamışsanız, artık beyaz tavşanı takip edip harikalar diyarına yuvarlanmaya hazırsınız. Ya da mavi hapı alıp normal yaşantınıza dönebilirsiniz...

Koordinat Sistemleri ve Vektörler

Vektör efektlerinin temelini oluşturan kavram çok boyutlu uzay geometrisidir. Bu karizmatik ada kanmayın aslında basit bir konudur. Bizim üzerinde çalışacağımız koordinat sistemleri birbirine dik eksenlerin oluşturduğu sistemler olacak. İlk olarak bu koordinat sistemlerinde yer alan vektörler ve onların üzerindeki basit bazı işlemlerden bahsedeceğiz. Ardından vektörler üzerinde dönüşüm adı verilen işlemlerden bahsedeceğiz.

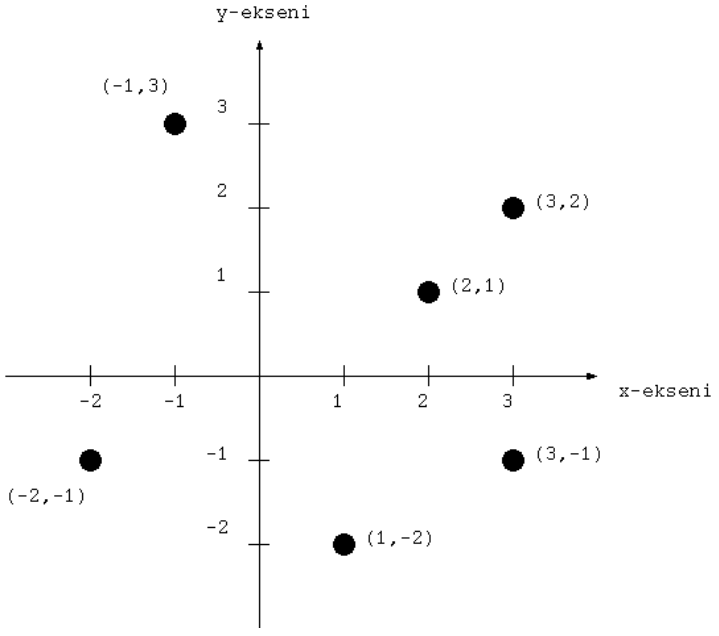
Dönüşümler, yer değiştirme, döndürme ve ölçekleme olacak. Bu kavramları önce 2 boyutlu uzayda anlatacağım. Ardından 3 boyuta geçmemiz zor olmayacak.

İki boyutlu uzay Şekil 1'de görüldüğü gibi birbirine dik x ve y eksenlerinin ifade ettiği bir uzaydır. Bu uzayda her nokta bir x ve bir y koordinatı ile ifade edilebilir.



Şekil-1: 2 boyutlu uzayda eksenler

Bu uzayda noktaların x ve y koordinatları ile nasıl ifade edildiğine Şekil 2'deki örneklerden bakabilirsiniz. Koordinat sisteminde (0,0) noktası özel olarak adlandırılır. Bu noktaya **origin** denir.



Şekil-2: örnek noktalar

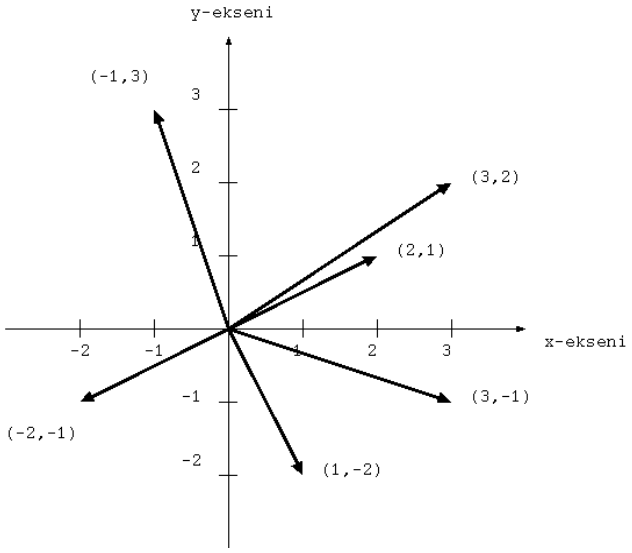
Vektörler

Vektörler matematikte çok boyutlu değerlerdir. Bu ne demek? Matematikte büyüklükler skaler ve vektörel diye iki gruba ayrılır. Skaler değerler 3, 5, 12000, $4\sqrt{2}$ gibi tek bir reel sayı ile ifade edilebilen değerlerdir. Vektörler ise çok boyutlu değerlerdir. Çok boyutlu bir uzayda bir yön ve büyüklük ifade ederler.

Vektörler en çok fizikte kuvvetleri, manyetik ve elektriksel alanları ifade etmekte kullanılırlar. Vektörleri çok boyutlu sayılar gibi düşünürseniz onlar üzerinde toplama çıkarma gibi işlemlerin de yapılabileceğini anlarsınız.

Şekil-2'deki noktaları vektörler olarak da düşünebiliriz. Vektörler şemalarda gösterilirken bir ok olarak gösterilirler ve V1, V2, P, Q gibi büyük harf ve rakamlarla isimlendirilirler. Ok olan uca baş, diğer uca kuyruk denir

Şekil-3'te örnek vektörler görebilirsiniz. Unutmamanız gereken bir nokta var. Vektörleri ifade eden oklar koordinat sisteminde farklı yerlere çizilebilirler. Önemli olan vektörün kuyruğu origine getirilince vektörün başının nereye geldiğidir. Vektörün değeri, kuyruk origindeyken baş noktasının koordinatlarıdır.



Şekil-3: Örnek vektörler

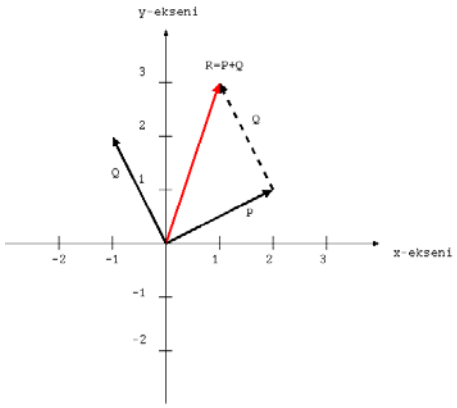
Vektörlerin Toplanması

Vektörlerde toplama işlemi şöyle yapılır. Diyelim ki P(2,1) ile Q(-1,2) vektörlerini toplamak istiyoruz. Bütün yapmamız gereken aynı eksenlerin koordinatlarını toplamaktır. Yani iki vektörün x koordinatları olan 2 ve -1 değerlerini toplayarak sonuç vektörünün x koordinatını elde ederiz. Aynı işlemi y koordinatlarına da yaparız.

P + Q = R
(2,1) + (-1,2) = (2+(-1), 1+2) = (1,3)

yani genel olarak:
(x1,y1) + (x2,y2) = (x1+x2, y1+y2)

Vektör toplamasını aynı zamanda görsel olarak da yorumlayabiliriz. P vektörünü kuyruğu origine gelecek şekilde yerleştirdikten sonra Q vektörünü kuyruğu P'nin başına gelecek şekilde yerleştiririz. Bu anda Q'nun başı nereye geliyorsa sonuç vektörü originden bu noktaya kadar ki vektördür. (Bkz. Şekil 4)



Şekil-4: Vektörlerin Toplanması

Vektörlerin Tersi

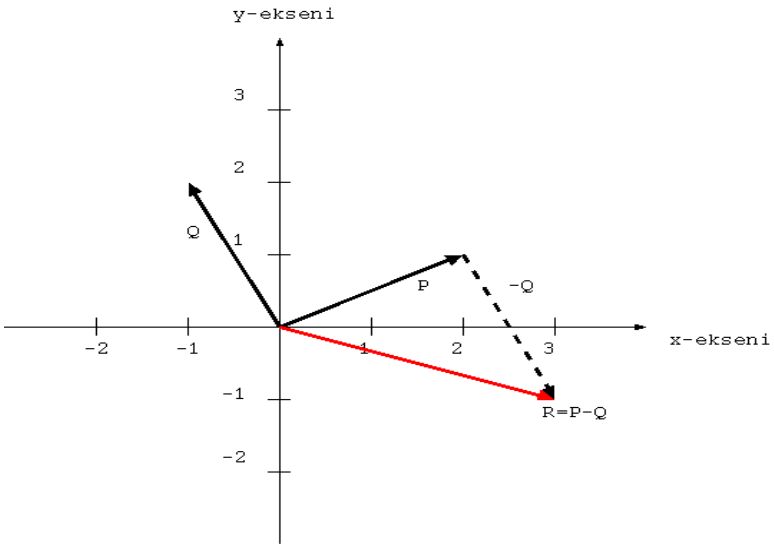
Vektörlerin tersini almak için bütün yapmamız gereken koordinatların tersini almaktır.
-(P(x1,y1)) = (-x1,-y1)

Grafiksel olarak bunun anlamı vektörün başı ve kuyruğunu yer değiştirmektir. Yani vektörü tam ters istikamete baktırmaktır.

Vektörlerin Çıkarılması

Vektörlerin çıkarılması demek birinci vektör ile ikinci vektörün tersinin toplanması demektir. Başka bir deyişle koordinatlar arası çıkarma işlemi yapılır.
P - Q = P + (-Q) = R
(x1,y1) - (x2,y2) = (x1-x2,y1-y2)

Grafik yorumu ise şudur. İkinci vektörün tersinin kuyruğu birinci vektörün başına eklenir.(Bkz Şekil-5)



Şekil-5: Vektörlerin Çıkarılması

Bu sayı için bahsedeceğimiz konular bu kadar. Önümüzdeki sayıda vektörel dönüşümlerle devam edeceğiz. O zamana kadar hoşça kalın.

HADES' BASIC

HAZIRLAYAN : İSMAİL ŞAHİN
İLETİŞİM : HADES@HI3S.ORG HADES6510@YAHOO.COM
NEREDEN : İSTANBUL
NİCK / GRUP(LAR) : HADES / HI3S
WEB SİTE : WWW.HI3S.ORG HTTP://HADES6510.SITEMYNET.COM

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

Herkese tekrar merhaba. Geçen sayımızda C64'ün yetersiz olan Basic komutlarına yeni komutların nasıl eklendiğini görmüş ve 4 tanede yeni komut eklemiştik. Bu sayımızda da C64'e işe yarar birkaç komut daha ekliyoruz. Yeni komutlarımızın 9 tanesi Sprite ile ilgili, 1 tanesi ise kursörü ekranda istediğiniz yere yerleştirmenizi sağlayan "CURSOR" komutu. Bildiğiniz gibi "SPRITE" C64'ün yazı ve grafiklerinden bağımsız olan, programlanabilen ve çeşitli özellikleri olan hardware tabanlı grafik elemanlarıdır. Oyunlarda yönettiğiniz kahramanlar, araçlar, düşmanlar, hareketli şekiller hep sprite'tır. C64'ün oyun bolluğu bu sprite'ların sayesindedir. Buraya kadar herşey güzel ama C64'ün "her güzelin bir kusuru vardır" sözünü doğrulayan bir kusuru vardır. Gerek sprite, gerek ses olsun C64'ün bu özelliklerini kullanabilmenizi sağlayan hiç bir komutu yoktur. Ekranda bir sprite gösterebilmek için en az 4 tane POKE komutu kullanmanız gerekmektedir. Yeni komutlarımız sizleri POKE komutu yazmaktan kurtaracaktır. Komutların kullanımı çok basittir. Aslında komutlar bu kadar fazla değildi. Bazı komutların parametreleri opsiyonel durumdaydı, yani kullanma zorunluluğu yoktu. Bu parametrelerin ayrı bir komut altında kullanılabilmesi için değişiklik yaptım.

Sprite komutlarımızın neler olduğuna geçmeden önce birazda "CURSOR" komutundan bahsedelim. C64'ün basic komutları arasında maalesef kursörle ilgili bir komut bulunmamaktadır. Ekranın her hangi bir yerine birşeyler yazmak istediğinizde tırnak işaretleri arasında bol miktarda –kursör aşağı ve kursor sağa- karakterlerini kullanmanız gerekmektedir. Yeni komut ile bunların hiç birisine gerek kalmayacak. Mesela CURSOR 10,20:PRINT "C64 TURKIYE" ile 10.satır 20.sütundan itibaren "C64 TURKIYE" yazısını yazmış oluyorsunuz.

Konuyu daha fazla dağıtmadan SPRITE komutlarımızı açıklayalım.

1- **SPRITE sprnum, açık/kapalı** : Bu komut ile istediğiniz sprite'ı açıp kapatabilirsiniz. Sprite'ın ekranda gözükmesi için koordinatlarının uygun sınırlar arasında olması gereklidir. Mesela 24,20 (x,y) koordinatlarında bulunan bir sprite gösterilmeyecektir. (Laf aramızda eğer alt ve üst BORDER'leri ortadan kaldırırsanız Y koordinatı sorun olmaz ☺)
2- **SPRPOS sprnum, x-koor, y-koor** : Yukarıda bahsettiğimiz koordinatları bu komutla girebilirsiniz. Bir sprite'ın ekranda tam olarak gözükeceği ilk koordinatlar x=24 y=50 dir.
3- **SPRPNT sprnum, göstergeç** : Sprite'ın şekil datalarının alınacağı hafıza adresini VIC'e bildirmek için kullanılır. Bu komut default pointer adresleri olan $07F8-$07FF arasını kullanır. Eğer VIC BANK ve/veya ekran belleğinin yeri değiştirilmişse sprite'ın şekli bozuk olacaktır. Aslında VIC BANK ve ekran belleği kontrolu yapan ve adresleri ayarlayan bir rutin programa eklenebilir.
4- **SPREXP sprnum, x-genişletme, y-genişletme** : Bu komut ile sprite'ınızın X ve Y eksenlerinde 2 kat genişleme özelliğini ayarlayabilirsiniz
5- **SPRMUL sprnum, açık/kapalı** : Hangi sprite'ın multicolor modda görüneceğini bu komut ile ayarlayabilirsiniz.
6- **SPRMCL renk1, renk2** : Bu komutla multicolor sprite renklerini girebilirsiniz.

7- **SPRCOL sprnum, renk :** Sprite'ların rengini bu komutla değiştirebilirsiniz.
8- **SPRIOR sprnum, öncelik :**Bu komut ile sprite'ların normal yazı karakterlerinin önünde mi yoksa arkasında mı gözükeceğini belirleyebilirsiniz.
9- **SPRALL açık/kapalı :** Bu komut ise bütün sprite'ları açabilir veya kapatabilirsiniz. Bu komuttan önce bütün sprite'ların x ve y koordinatları, göstergeçleri vs. girilmiş olması gerekmektedir.
10- **CURSOR satır, sütun :** Bu komut ile artık PRINT komutuyla yazıyı ekranda istediğiniz yere rahatça yazdırabileceksiniz.

Şimdide gelelim komutlardaki parametrelerin aldıkları değerlere...

SPRNUM : Sprite numarası demektir ve 0...7 arasında olması gerekmektedir. 8 ve yukarısı değerler hata verir.

AÇIK / KAPALI : Bu parametre ile ilgili özelliğin açık veya kapalı olması seçilir. 0: Kapalı 1: Açık demektir. 2 ve yukarısı değerler hata verir.

X-KOOR : Sprite'ın ekrandaki pozisyonunun X koordinat değeridir. 0...65535 arasında bir değer alabilir. 65536 ve yukarısı değerler hata verir. 255'ten büyük X koordinatları için Extra X-koordinat biti otomatik olarak ayarlanır. Bu durumda göre bir sprite'ın alabileceği X-koordinat değeri en fazla 511 olabilir. Bir sprite'ın ekranda tam olarak görüntülenebileceği minimum X-koordinat değeri 24'tür. Eğer sprite X ekseninde genişletilmemiş ise ekranda tam olarak görüntülenebileceği maksimum X koordinat değeri ise 320'dir. 320'den itibaren sprite görüntüden çıkmaya başlar. Eğer OPEN SIDE BORDER demo tekniğini kullanırsanız 320'den sonraki koordinatlarda da sprite görünüyor olacaktır.

Y-KOOR : Sprite'ın ekrandaki pozisyonunun Y koordinat değeridir. 0...255 arasında olmalıdır. 256 ve yukarısı değerler hata verir. Normalde bir sprite'ın ekranda tam olarak görüntülenebileceği minimum Y koordinat değeri 50'dir.

GÖSTERGEÇ : Bir sprite'ın şekil datalarının tutulduğu bellek adresleri göstergeç ile VIC'e bildirilir. 0...255 arasında olmalıdır. 256 ve yukarısı değerler hata verir. GÖSTERGEÇ * 64 = ŞEKİL DATA ADRESİDİR. Bu komut sadece VICBANK 0'da ve EKRAN BELLEĞİNİN YERİ DEĞİŞMEDİYSE geçerlidir. İsteyen source code'a VIC bank ve ekran belleği adres kontrolü ekleyebilir.

X-GENİŞLETME, Y-GENİŞLETME : Sprite'ın X ve/veya Y eksenlerinde 2 kat büyümesini kontrol edebilirsiniz. 0: Genişleme yok, 1: Genişleme var demektir. 2 ve yukarısı değerler hata verir.

RENK, RENK1, RENK2 : 0...255 arasında olmalıdır. 256 ve yukarısı değerler hata verir. İlk 0...15 arası renk değerleri her 16 sayıda tekrar eder. Yani herhangi bir renk değerine 16 eklerseniz yine aynı rengi elde edersiniz.

ÖNCELİK : Bu parametre ile bir sprite'in ekrandaki bir karakterin önünde veya arkasında olmasını sağlayabilirsiniz. 0 ise sprite önde, 1 ise arkada gözükecektir. 2 ve yukarısı değerler hata verir.

SATIR, SÜTUN : Kursörün ekranda gözükeceği satır ve sütun değerini bildirmeye yarar. Satır değeri 0...24, sütun değeri 0...39 arasında olmalıdır. Satır için 25 ve yukarısı, sütun için 40 ve yukarısı değerler hata verir.

Yukarıdaki açıklamalardan sonra C64'te sprite kullanmak artık çok basit olacaktır. Daha önce bir "sprite editör" ile sprite'larınızı hazırlayın. Sonra sprite komutlarıyla denemeler yapın. Bir gecikme rutini (işte size bir BASIC komutu daha : **DELAY süre** ile gecikme yaptırabiliriz.) SPRPNT komutuyla bir sprite'ın datalarının alınacağı adresi değiştirirseniz basit bir animasyon yapmış olursunuz. Programın assembler listesinin sonunda iki adet basic program listesi göreceksiniz. Birincisi C64'ün poke komutlarıyla sprite hareket ettirirken ikincisinde bu iş için yeni komutları kullanacağız. Bu arada bazı temel basic komutlarını bildiğinizi varsayıyorum. Mesela PRINT, FOR-NEXT vs..

```
                *=$c000

                lda   #<yeni
                ldx   #>yeni
                sta   $0308
                stx   $0309
                rts

yeni            lda   $7a
                ldx   $7b
                sta   $fb
                stx   $fc
                lda   #<comset
                ldx   #>comset
                sta   $fd
                stx   $fe
                ldx   #$00
                ldy   #$00
loop0           jsr   $0073
                sta   $02
loop1           lda   ($fd),y
                cmp   $02
                bne   nextcom
                iny
                tya
                cmp   comlen,x
                bne   loop0
exec            txa
                asl
                tax
                lda   comexe+1,x
                pha
                lda   comexe,x
                pha
                rts
nextcom         lda   comlen,x
                clc
                adc   $fd
                sta   $fd
                bcc   next0
                inc   $fc
next0           inx
                cpx   commax
                bne   loop1
                lda   $fb
                ldx   $fc
                sta   $7a
                stx   $7b
                jmp   $a7e4
;-------------------------------------------------------------------
commax          .byte 10
comlen          .byte 6,4,6,4,6,6,6,5,6,5

comset          .text "sprite"
                .text "spr"
                .byte $b9         ;POS
                .text "sprpnt"
                .text "spr"
                .byte $bd         ;EXP
                .text "sprmul"
                .text "sprmcl"
```

```
                .text "sprcol"
                .text "spri"
                .byte $b0         ;OR
                .text "sprall"
                .text "curs"
                .byte $b0         ;OR

comexe          .word sprite-1
                .word sprpos-1
                .word sprpnt-1
                .word sprexp-1
                .word sprmul-1
                .word sprmcl-1
                .word sprcol-1
                .word sprior-1
                .word sprall-1
                .word cursor-1
;-------------------------------------------------------------------
sprite          jsr   spritex
                bcs   spron
                and   $d015
                jmp   cont1
spron           ora   $d015
cont1           sta   $d015
                jmp   $a7ae
;-------------------------------------------------------------------
sprpos          jsr   comspr
                jsr   $aefd
                jsr   $b7eb
                stx   spry
                ldx   sprnum
                lda   $15
                beq   noext
                lda   on,x
                ora   $d010
                jmp   cont2
noext           lda   off,x
                and   $d010
cont2           sta   $d010
                txa
                asl
                tax
                lda   $14
                sta   $d000,x
                lda   spry
                sta   $d001,x
                jmp   $a7ae
;-------------------------------------------------------------------
sprpnt          jsr   comspr
                jsr   common
                txa
                ldx   sprnum
                sta   $07f8,x
                jmp   $a7ae
;-------------------------------------------------------------------
sprexp          jsr   spritex
                bcs   sprex
                and   $d01d
                jmp   cont3
sprex           ora   $d01d
cont3           sta   $d01d
```

```
                jsr   common
                jsr   spritex2
                bcs   sprey
                and   $d017
                jmp   cont4
sprey           ora   $d017
cont4           sta   $d017
                jmp   $a7ae
;---------------------------------------------------------------------
sprmul          jsr   spritex
                bcs   multi
                and   $d01c
                jmp   cont5
multi           ora   $d01c
cont5           sta   $d01c
                jmp   $a7ae
;---------------------------------------------------------------------
sprmcl          jsr   $b79b
                stx   $d025
                jsr   common
                stx   $d026
                jmp   $a7ae
;---------------------------------------------------------------------
sprcol          jsr   comspr
                jsr   common
                txa
                ldx   sprnum
                sta   $d027,x
                jmp   $a7ae
;---------------------------------------------------------------------
sprior          jsr   spritex
                bcs   pri
                and   $d01b
                jmp   cont6
pri             ora   $d01b
cont6           sta   $d01b
                jmp   $a7ae
;---------------------------------------------------------------------
sprall          jsr   $b79b
                jsr   spritex2
                bcs   allon
                lda   #$00
                .byte $2c
allon           lda   #$ff
                jmp   cont1
;---------------------------------------------------------------------
cursor          jsr   $b79b
                cpx   #$19
                bcs   error
                stx   line
                jsr   common
                cpx   #$28
                bcs   error
                stx   column
                ldx   line
                ldy   column
                jsr   $e50c
                jmp   $a7ae
error           jmp   $b248
;---------------------------------------------------------------------
spritex         jsr   comspr
```

```
                jsr   common
spritex2        cpx   #$02
                bcs   error
                cpx   #$01
                beq   onspr
offspr          ldx   sprnum
                lda   off,x
                rts
onspr           ldx   sprnum
                lda   on,x
                sec
                rts
;-------------------------------------------------------------------
comspr          jsr   $b79b
                cpx   #$08
                bcs   error
                stx   sprnum
                rts
;-------------------------------------------------------------------
common          jsr   $aefd
                jsr   $b79e
                rts
;-------------------------------------------------------------------
spry            .byte 0
sprnum          .byte 0
on              .byte 1,2,4,8,16,32,64,128
off             .byte 254,253,251,247,239,223,191,127
line            .byte 0
column          .byte 0

                .end
```

C64 BASIC KOMUTLARI	HADES BASIC KOMUTLARI
10 PRINT CHR$(147):REM EKRANI SIL 15 POKE 53280,6:POKE 53281,0:POKE 646,1 20 PRINT:PRINT REM KURSORU 2 SATIR ASAGI AL. (KURSOR 4.SATIRDA) 25 PRINT " C64'TE SPRITE KULLANMAK." 30 FOR F=0 TO 64:POKE 832+F,255:NEXT 40 POKE 2040,13:REM 632/64 50 POKE 53264,0 60 POKE 53287,7 70 POKE 53269,1 80 X=24:Y=50 90 POKE 53249,Y 100 X=X+1:IF X<256 THEN POKE 53248,X:GOTO 90 110 POKE 53264,1:X=24 120 POKE 53248,X 130 X=X+1:IF X<90 THEN 120 140 GOTO 50	10 SYS 58692:**REM CLS KOMUTU** 15 POKE 53280,6:POKE53281,0:POKE 646,1:**REM BORDER 6:REM PAPER 0:REM INK 1** 20 CURSOR 3,8: REM KURSOR 4. SATIRDA 25 PRINT "C64'TE SPRITE KULLANMAK." 30 FOR F=0 TO 64:POKE 832+F,255:NEXT 40 SPRPNT 0,13 50 REM 60 SPRCOL 0,7 70 SPRITE 0,1 80 X=24:Y=50 90 SPRPOS 0,X,Y 100 X=X+1:IF X<346 THEN 90 110 GOTO 80

Gördüğünüz gibi program hem kısalmakta hemde basitleşmektedir. C64'ün basic komutlarında bir sürü adres bilmeniz gerekirken yeni komutlarda bunlara gerek kalmamaktadır. Bu arada zaman yetersizliğinden dolayı assembler listesinde açıklama yapamadım. Anlamadığınız yerler olursa foruma yazabilirsiniz.

PROGRAM BÖLÜMÜ

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

ÇİZGİ KURSÖR

HAZIRLAYAN : İSMAİL ŞAHİN
İLETİŞİM : HADES@HI3S.ORG HADES6510@YAHOO.COM
NEREDEN : İSTANBUL
NİCK / GRUP(LAR) : HADES / HI3S
WEB SİTE : WWW.HI3S.ORG HTTP://HADES6510.SITEMYNET.COM

Eğer C64'ün yanıp sönen kare şeklindeki kursöründen sıkıldıysanız aşağıdaki kısa programla (sadece 96 byte uzunluğundadır) kursörünüz artık PC'lerin DOS zamanındaki kursörü şeklinde olacaktır. Peki bu değişiklik nasıl oluyor? Yeni kursörümüz aslında bir sprite'dır ve interrupt tekniği kullanılmıştır. Normal kursörün satır ve sütun değerleri okunarak programda sprite'ın X ve Y koordinat değerlerine dönüştürülmektedir.

```
                *=$0801

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

                sei                 ;interruptları durdur
                lda   #<cursor      ;programın low byte'ını
                sta   $0134         ;interrupt vektörünün low byte'ına yaz
                lda   #>cursor      ;programın high byte'ını
                sta   $0315         ;interrupt vektörünün high byte'ına yaz
                ldx   #$00          ;X registerini sıfırla
                txa                 ;Akü'ye X registerinin değerini yükle
loop0           sta   $03c0,x       ;Sprite data adresine 0 yaz
                inx                 ;sayacı 1 arttır
                cpx   #$40          ;#$40 (64) oldumu ?
                bne   loop0         ;olmadıysa tekrarla
                lda   #$ff          ;çizgi şekli olması için $ff sayısını
                sta   $03c0+21      ;sprite'ın uygun data adresine yaz
                lda   #$03c0/64     ;sprite data başlangıcını
                sta   $07f8         ;1. sprite pointerine yaz
                lda   #$01          ;1. Sprite'ı
                sta   $d015         ;görünür yap
                cli                 ;interruptları başlat
                rts                 ;programı bitir

cursor          lda   $d3           ;normal kursörün sütun pozisyonunu oku
                cmp   #$1d          ;$1d (19) ile karşılaştır
                bcc   no_extra      ;$1d'den küçükse git
                ldx   #$01          ;sprite'ın extra x koordinatı var
                .byte $2c           ;bilgisayarı kandırma sayısı
no_extra        ldx   #$00          ;sprite'ın extra x koordinatı yok
                stx   $d010         ;spritelar için extra x koordinat adresi
```

```
asl             ;sütun değerini 2 ile çarp
asl             ;sonucu 2 ile çarp
asl             ;önceki sonucu 2 ile çarp
clc             ;"cary" bitini sıfırla
adc   #$18      ;$18 (24) ile topla
sta   $d000     ;1. sprite'ın X koordinat adresine yaz.
```

Burada biraz bilgi verelim. Yukarıda 3 defa asl komutunu kullandık. Peki neden böyle bir işlem yaptık? Normalde kursörün sütun değerini olduğu gibi sprite'ın x koordinatına veremeyiz. Eğer verirsek sprite'ımız sadece 1 pixel ilerlemiş olur. Bir karakter normalde 8 pixel genişliğindedir. Kursör dediğimiz şekil aslında bir karakterdir ve reverse space karakterinden başka bir şey değildir. Kursörün bir sonraki sütuna gitmesi demek sprite'ımızında 8 pixel ilerlemesi demektir. Kursörün sütun değeri*8 sayısı bize sprite'ın X koordinatını verecektir. Assemblerde bir sayıyı 8 ile çarpmak, o sayıyı 3 defa sola kaydırmak yani 3 defa ASL komutu kullanmak demektir. Buraya kadar tamam. Peki $18 (24) sayısı nereden geliyor. Bir sprite'ın ekranda tamamen görünür halde olması için X koordinatı $18 olmalıdır. Bu sayıyı 3 ASL komutu sonunda elde ettiğimiz sayıyla toplarsak işimiz tamam olacaktır.

```
lda   $d6       ;kursörün satır değerini oku
asl             ; 2 ile çarp
asl             ; bir daha 2 ile çarp
asl             ; son kez 2 ile çarp
clc             ;"carry" bitini sıfırla
adc   #$32      ;$32 (50) ile topla
sta   $d001     ;1. sprite'ın Y koordinat adresine yaz.
```

Yukarıdaki açıklamanın bir benzerini satır değeri için yapıp sonucu sprite'ın Y koordinat adresine yazdığımızda kursörün o anki pozisyonunda birde sprite göstermiş oluruz.

```
inc   $d027     ;1. sprite'ın rengini değiştir.
jsr   $ffea     ;zamanı ilerlet ve stop tuşunu kontrol et.
jmp   $ea61     ;Romdaki interrupt rutinine geri dön.

.end
```

Geldik son açıklamaya. Kendi yazdığımız bir interrupt rutininin çalışması bitince son komutun normalde JMP $EA31 veya $EA81 olması gerekir. $EA81 deki komutlar $EA31'den başlayan komutların devamıdır. Peki aradaki $50 (80) byte'ta neler olmaktadır? İlk önce Stop tuşu kontrolu ve zaman ilerletme yapılıyor. Sonra kursörün yakılıp söndürülmesi ve kursörle ilgili karakter ve renk işlemleri yapılıyor. Daha sonra ise teyp tuşlarına basılıp basılmadığı ve teyp motor on / off işlemleri yapılıyor. En son olarak ise klavye kontrolu yapılmakta ve $EA81 adresine gelinmektedir. $EA81'den itibaren ise sırayla Y, X ve A registerleri yığından geri alınıp interrupt rutini bitirilmektedir.

Bizim rutinimizde ise önce JSR $FFEA ile bir işlem yapılmakta ve JMP $EA61 ile çıkış yapılmaktadır. Eğer JMP $EA31 kullansak kursör işlemleri yapılacak, JMP $EA81 kullansak tuşlar çalışmaz hale gelecek. Bu nedenle önce JSR $FFEA ile stop tuşunun kontrol edilmesini sağlıyoruz, sonrada kursör işlemlerine hiç uğramadan geri kalan işler için JMP $EA61 komutuyla işimizi bitiriyoruz.

C64 İÇİN BAŞ AŞAĞI KARAKTER SETİ PROGRAMI

HAZIRLAYAN : **İSMAİL ŞAHİN**
İLETİŞİM : HADES@HI3S.ORG HADES6510@YAHOO.COM
NEREDEN : **İSTANBUL**
NİCK / GRUP(LAR) : **HADES / HI3S**
WEB SİTE : WWW.HI3S.ORG HTTP://HADES6510.SITEMYNET.COM

```
                *=$0801

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

                sei                 ;İnterruptları engelle
                lda   $01           ;Karakter ROM'u
                and   #$fb          ;artık CPU tarafından
                sta   $01           ;okunabilir
                lda   #$d0          ;ROM adresi = $D000
                ldx   #$30          ;RAM adresi = $3000
                ldy   #$00          ;olacak şekilde ayarla.
                sty   $fb           ;$FB ve $FC adreslerinde
                sta   $fc           ;ROM adresi tutuluyor.
                sty   $fd           ;$FD ve $FE adreslerinde
                stx   $fe           ;RAM adresi tutuluyor.
                ldx   #$10          ;Blok sayacı = $10 (16)
transfer        lda   ($fb),y       ;ROM'dan o anki adresi
                sta   ($fd),y       ;oku ve RAM'e kopyala
                iny                 ;byte sayacını 1 arttır
                bne   transfer      ;"0" değilse geri dön
                inc   $fc           ;ROM HIGH adresi 1 arttır
                inc   $fe           ;RAM HIGH adresi 1 arttır
                dex                 ;Blok sayacını 1 azalt
                bne   transfer      ;"0" olmadıysa geri dön.
                lda   $01           ;I/O bölgesi eski haline
                ora   #$04          ;getiriliyor. CPU artık
                sta   $01           ;karakter roma erişemez
                cli                 ;interruptlara izin ver
                lda   #$1c          ;VIC'e yeni karakter
                sta   $d018         ;setinin yerini bildir.
new_fonts       lda   #$30          ;Karakter seti başlangıç
                ldy   #$00          ;adresi $3000 olarak
                sty   $fb           ;$FB ve $FC adreslerine
                sta   $fc           ;kaydediliyor
loop2           ldy   #$00          ;Y registeri byte sayacı=0
loop0           lda   ($fb),y       ;O anki adresten bir byte oku
                pha                 ;okunan değeri yığına at
                iny                 ;sayacı 1 arttır
                cpy   #$08          ;8 oldumu ?
                bne   loop0         ;olmadıysa bir byte daha oku
                ldy   #$00          ;Y registeri byte sayacı=0
loop1           pla                 ;yığındaki değeri al
                sta   ($fb),y       ;O anki adrese yaz
                iny                 ;sayacı 1 arttır
                cpy   #$08          ;8 oldumu ?
                bne   loop1         ;olmadıysa yeni değeri al.
```

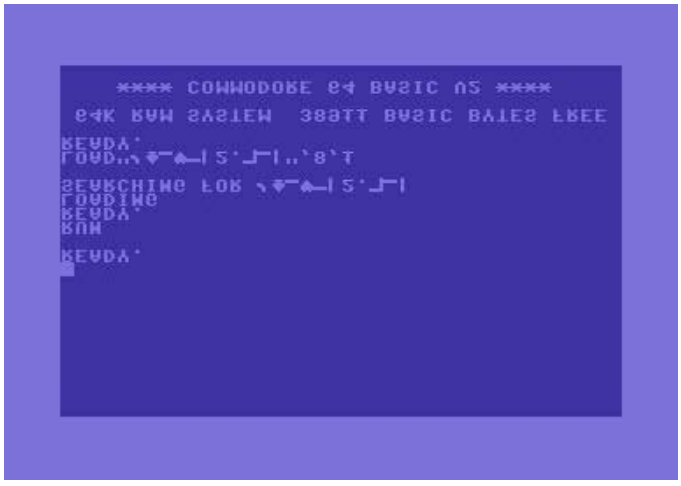
Yukarıdaki komutlarla bir karakteri baş aşağı çeviriyoruz. Burada asıl can alıcı nokta "YIĞIN" dediğimiz bölgenin kullanılmasıdır Yığın "LIFO" -Last In First Out- mantığıyla çalışır. Yani yığına atılan en son sayı ilk önce geri alınır. Bizim komutlar ilk çalışmaya başladığında önce $3000'den itibaren ilk 8 byte'ı okuyup yığına atmaktadır. Böylece yığına en son $3007 adresindeki değer atılmış olmaktadır. Yığından geri alma işlemi sırasında ise ilk önce $3007 adresinin değeri alınmaktadır. Bu işlemi 8 defa tekrarladığımızda ise bir karakterin datalarını kendi içinde tersten yerleştirmiş oluyoruz. Aşağıdaki tablo daha açıklayıcı olacaktır.

$3000 --> yığın --> $3007
$3001 --> yığın --> $3006
$3002 --> yığın --> $3005
$3003 --> yığın --> $3004
$3004 --> yığın --> $3003
$3005 --> yığın --> $3002
$3006 --> yığın --> $3001
$3007 --> yığın --> $3000

```
                lda   $fb         ;Karakter setinin LO adresini al
                clc               ;Elde bitini sfırla
                adc   #$08        ;8 ile topla
                sta   $fb         ;LO adrese geri yaz.
                bcc   next_char   ;Toplama sonucu "elde" biti "0" ise git
                inc   $fc         ;Karakter setinin HI adresini 1 arttır
next_char       lda   $fc         ;HI adresi al
                cmp   #$40        ;#$40 oldumu? Olmadıysa bir sonraki
                bne   loop2       ;karakter için işlemleri tekrarla
                rts               ;Olduysa programı bitir.

                .end
```

Gördüğünüz gibi kısacık bir programla $3000-$4000 arasındaki karakter setimizi baş aşağı çevirmiş olduk. Programın listesini “program dökümleri” sayfasında bulabilirsiniz. Ekran görüntülerimiz ise aşağıdadır.



Normal karakter setli ekran görüntüsü **Ters karakter setli ekran görüntüsü.**

Bir başka karakter setinde görüşmek dileğiyle.

ROM RUTİNLERİ

HAZIRLAYAN : İSMAİL ŞAHİN
İLETİŞİM : HADES@HI3S.ORG HADES6510@YAHOO.COM
NEREDEN : İSTANBUL
NİCK / GRUP(LAR) : HADES / HI3S
WEB SİTE : WWW.HI3S.ORG HTTP://HADES6510.SITEMYNET.COM

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

ROM rutinlerini tanıtmaya devam ediyoruz.
İlk adresimiz $A3BF (41919).

ADRES : $A3BF (41919)
GÖREVİ : Data blok taşıma.

Bu adresi bellekteki bir bölgeyi başka bir yere taşıma amaçlı olarak kullanabilirsiniz. Zero Page'de bulunan 6 adet adresi kullanıp istediğiniz uzunluktaki bir bölgeyi taşıyabilirsiniz. Şimdi bir örnek ile bu adresin nasıl kullanıldığını öğrenelim. Örneğimizde $c000-$c750 arasındaki bölgeyi $2600-$2d50 arasına taşıyalım.

```
                *=$0801

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

transfer1       lda   #$00          ;Başlangıç adresi LOW byte
                ldx   #$c0          ;Başlangıç adresi HIGH byte
                sta   $5f           ;Başlangıç LOW byte pointer
                stx   $60           ;Başlangıç HIGH byte pointer
                lda   #$50          ;Bitiş adresi LOW byte
                ldx   #$c7          ;Bitiş adresi HIGH byte
                sta   $5a           ;Bitiş LOW byte pointer
                stx   $5b           ;Bitiş HIGH byte pointer
                lda   #$50          ;Hedef bitiş adresi LOW byte
                ldx   #$2d          ;Hedef bitiş adresi HIGH byte
                sta   $58           ;Hedef bitiş LOW byte pointer
                stx   $59           ;Hedef bitiş HIGH byte pointer
                jsr   $a3bf
                rts
```

Transfer işlemini yapan rutin 28 byte yer kaplamaktadır.
Bu adresin kullanılışı şöyledir. Kopyalayacağınız bölgenin başlangıç adresi $5f-$60 adreslerine, bitiş adresi $5a-$5b adreslerine, yeni bölgenin bitiş adresi ise $58-$59 adreslerine yazılır ve rutin çağrılır. Bu rutini kullanacağınız zaman BASIC ROM'un devrede olması gereklidir.

Aynı taşıma işlemini birde başka rutin yazarak yapalım. Taşınacak bölgenin uzunluğu $c750-$c000=$0750 byte olup 8 döngüde rahatça transfer edebiliriz.

```
transfer2       ldx   #$08        ;Döngü sayacı
                ldy   #$00        ;Byte sayacı
kaynak          lda   $c000,y     ;Kaynak adresi
hedef           sta   $2600,y     ;Hedef adresi
                iny
                bne   kaynak
                inc   kaynak+2
                inc   hedef+2
                dex
                bne   kaynak
                rts
```

Transfer2 rutinimiz 23 byte uzunluğundadır. Yani üstteki rutinden 5 byte daha kısadır. Ama bana göre 2 dezavantajı vardır. Birincisi : Transfer1 rutininde istediğiniz uzunlukta byte transfer edebilirken, transfer2 rutininde transfer edilen byte uzunluğu 256'nın katları şeklinde olacaktır. Transfer1 rutininde $0750 (1872) byte transfer edilirken, transfer2 rutininde 256'nın katları şeklinde olduğu için 2048 byte transfer edilecektir. Eğer hafıza düzenlemeniz biraz karışıksa fazladan transfer edilen 1 byte bile programınızın bozulmasına neden olabilir. Bununda çaresi var aslında. Önce $0750 byte'ın $0700 byte'ını transfer edersiniz, sonrada uygun $50 byte için işlem yaparsınız. Bu ise fazladan 9 byte'lık bir code demektir. İkinci dezavantaj ise, transfer2 rutinini bu şekliyle sadece bir kez kullanılabilir. Çünkü kaynak ve hedef adreslerinin high byte'ları her döngüde 1 arttırılmaktadır. Eğer rutini ikinci kez kullanmak istiyorsanız, rutinin başında kaynak ve hedef adreslerinin high bytle'rını program içindeki yerlerine yazmanız geerekir. Bu işlem ise 10 byte daha kullanmak demektir. Böylece TRANSFER1=28 byte, TRANSFER2=33 byte yer kaplamış olacak, ayrıca 1. dezavantaj yine devam edecektir.

ADRES : $E8EA (59626)
GÖREVİ : Ekranı bir satır yukarı kaydırma.

The anatomy of a COMMODORE 64 isimli kitapta SCROLL SCREEN olarak isimlendirilen bu rutin ekranı bir satır yukarı kaydırır. Normalde ekranın yukarı kayması için kursörün en alt satırda olması ve RETURN veya CURSOR DOWN tuşuna basılmış olması gerekir. Bu adres ise kursörün pozisyonunu dikkate almayıp ekranı olduğu gibi bir satır yukarı kaydırır. Peki bu adresle ne yapılabilir? Mesela Basic'e ROLL komutu ekleyip, Basic ile hazırladığınız bir açılış ekranını 1 satır yukarı kaydırıp, üstten kaybolan satırı en alt satırda tekrar çıkartabilirsiniz. Tabii bunun için gerekli olan programı yazmanız lazım. İsterseniz bu efekti ufak bir assembler rutiniyle yapmayı deneyelim.

```
                *=$0801

                .word nextline
                .word 2004        ; 2004
                .byte $9e         ; SYS
                .text "2061"      ; 2061
                .byte 0           ; komutu
nextline        .word 0
                ldx   #$00        ;Sayacı sıfırlayalım
loop0           lda   $0400,x     ;Ekranın ilk satırındaki
                sta   temp0,x     ;bilgileri kopyalayalım
                lda   $d800,x     ;Aynı işi renk belleği
                sta   temp1,x     ;için de yapalım.
                inx               ;sayacı 1 arttır
                cpx   #$28        ;40 oldu mu?
                bne   loop0       ;olmadıysa tekrarla
                jsr   $e8ea       ;Olduysa ekranı 1 satır yukarı kaydır
```

```
                ldx   #$00        ;sayacı sıfırlayalım
loop1           lda   temp0,x     ;kopyaladığımız değerleri ekranın
                sta   $07c0,x     ;en alt satırına yazalım.
                lda   temp1,x     ;Aynı işi renk belleği
                sta   $dbc0,x     ;için de yapalım.
                inx               ;sayacı 1 arttır
                cpx   #$28        ;40 oldu mu?
                bne   loop1       ;olmadıysa tekrarla
son             rts               ;evet oldu. Öyleyse çıkalım.
temp0           =son+1
temp1           =temp0+40
                .end
```

Eğer bu efekti $EA8A adresini kullanmadan yapsaydık oldukça uzun bir program yazmak zorunda kalacaktık.

ADRES : $FFF0 (65520)
GÖREVİ : Kursör pozisyonunu ayarlama / okuma

Romda öyle rutinler vardır ki sizleri birçok işten kurtarır. Birazdan anlatacağımız bu rutinde bunlardan biridir. Bir program yazmaya başladınız ve ekranın 12. Satır 10. Sütun pozisyonundan başlayan 10 harflik bir yazı yazmanız gerekiyor. Bu işlemi normalde iki şekilde yapabilirsiniz. Birinci olarak ekranın ilgili pozisyonunun ekran belleği adresini öğrenip yazınızı ekran kodu olarak yazdırırsınız. İkincisi kursörü önce en üst köşeye alırsınız. Daha sonra "return" tuşunun kodunu kullanarak kursörü istediğiniz satıra getirirsiniz. Yazacağınız yazıya biraz "boşluk" karakteri ekleyerek istediğiniz pozisyona yazmış olusunuz. İkisinden birini tercih edin. Birinci seçenekte ekran belleğinin adresleriyle uğraşırsınız. İkinci seçenekte ise fazladan byte kullanarak programınızı uzatırsınız.

Aranızda "Başka bir alternatif yok mu ?" diye soranlar olabilir. Evet var. KERNAL ROM'da, işletim sistemi rutinleri için sıçrama tablosu denilen ve Rom'un son 128 byte'ında yer alan, direct ve indirect jump komutlarından oluşan bir bölge vardır. Bu bölgenin $FFF0 adresi tam bize göredir. Ekranın istediğiniz yerine bir yazı yazmak için yapmanız gereken işlemler şunlardır. Önce "carry" bitini sıfırlayın. Sonra X registerine satır numarasını, Y registerine süttun değerini yükleyin. En son olarak bir JSR komutuyla rutini çağırın. Böylece kursörün pozisyonunu ayarlamış olduk. Daha sonra ise geçen sayıda tanıtmış olduğumuz $AB1E adresini kullanarak yazıyı ekrana yazdırın. İsterseniz buraya kadar yazdıklarımızla ilgili örnekler verelim.

Yazımız 10. Satır 5. Sütunda olsun.
İlk önce doğrudan ekran adresi vererek bu yazıyı yazdıralım. Bu arada ufak bir not düşelim. Vereceğimiz örneklerde **satır ve sütun numaraları "0" dan başlamaktadır.** Program yazarken buna dikkat edin.

POZİSYON = EKRAN BELLEĞİ BAŞLANGIÇ + SATIR * 40 + SÜTUN formülüne göre adresi hesaplayalım.
EKRAN BELLEĞİ BAŞLANGIÇ ADRESİ = 1024
POZİSYON = 1024 + 10*40 + 5 = 1024 + 400 + 5 = 1429 ($0595)

Şimdi yazıyı yazdıralım.

```
                ldx   #$00
loop0           lda   text,x
                sta   $0595,x
                inx
                cpx   #$0b
                bne   loop0
                rts
text            .scrl "c64 turkiye"
```

İşin hesaplama kısmı olmasa her şey çok güzel olacak. Şimdide ikinci şıkkı deneyelim.

```
                jsr   $e544
                ldx   #$00
loop0           lda   text,x
                jsr   $ffd2
                inx
                cpx   #$1a
                bne   loop0
                rts
text            .byte $0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d
                .byte $20,$20,$20,$20,$20
                .text "c64 turkiye"
```

$0d sayısı return tuşunun, $20 sayısı ise "boşluk" karakterinin kodudur.

Şimdide $FFF0 adresini kullanarak yazıyı yazdıralım.

```
                clc             ;"carry" bitini sıfırla
                ldx   #$0a      ;satır değeri
                ldy   #$05      ;sütun değeri
                jsr   $fff0     ;kursörü ayarla

                lda   #<text
                ldy   #>text
                jsr   $ab1e
                rts
text            .text "c64 turkiye"
                .byte 0
```

Örnekleri karşılaştırarak hangisi daha kolayınıza geliyorsa onu kullanın.

Şimdide kursörün pozisyonunu okuyalım.

```
                sec             ;"carry" bitini "1" yap
                jsr   $fff0     ;rutini çağır.
                ...
                ...
                rts
```

Rutinden geri dönüşte kursörün satır değeri X registerinde, sütun değeri ise Y registerinde bulunur.

Bildiğiniz gibi C64'ün basic komutlarında kursörle ilgili bir komut yoktur. $FFF0 adresini kullanarak, kursörü istediğiniz pozisyona getirmek için "CURSOR" komutu ekleyebilirsiniz.

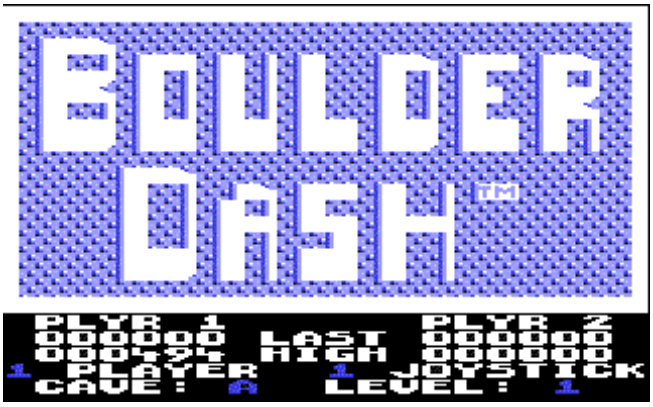
CURSOR 10,10 : PRINT "C64 TURKIYE" vs. kullanmak ne güzel olurdu değilmi?

(Bu yazıyı yazdıktan sonra "HADES'BASIC" bölümüne **"CURSOR"** komutunu ekledim.)

OYUN MERKEZİ

BOULDER DASH

HAZIRLAYAN : **YEŞİM GÜLEN**
İLETİŞİM : INSOMNIA@HI3S.ORG
NEREDEN : **ANKARA**
NİCK / GRUP(LAR) : **INSOMNIA / HI3S**
WEB SİTE : WWW.HI3S.ORG



Çoğumuzun saatlerce başından kalkamadığı elmasları toplamak uğruna can kaybettiği, Commodore 64'ün klasikleşmiş oyunlarından biriydi. Aslında basit grafiklerle hazırlanmış ama bol bulmaca dolu bu oyun bir bölümden diğerine geçmek için çabalarken sizi kolayca esir alıyor ve ertesi gün okula uykusuz gitmenize sebep oluyordu.

İlk olarak 1984'de First Star Software tarafından piyasaya sürülen oyun ardından büyük yankılar yapınca devamları, sahteleri ve benzerleri (şu an bende PC formatını indirdim ama commodore' da oynadığım zevki vermiyor nedense!) piyasaya sürüldü. Birçok ödül alan Boulder Dash oyunları halen üretiliyor ve First Star Software, Boulder Dash'in tüm yasal haklarını elinde bulunduruyor. İstatistiklere göre bugüne kadar 2 milyonun üzerinde oyun satılmış ve halen satılmakta. Boulder Dash oyunlarını ve gelişmelerini takip etmek istiyorsanız Rockford@firststarsoftware.com adresine mail atmanız yeterli.

1984'te yayınlanan birinci oyunun ardından çıkan Boulder Dash 2- Rockford's Revenge birleştiren EA Super Boulder Dash' i (1986-1987) piyasaya sürdü fakat 100,000 adetten fazla satışa ulaşamadı.

Hatırlamayanlar için, Rockford adındaki ufak adamı sağa, sola, yukarı ve aşağı hareket ettirerek mağaradaki elmasları topluyorsunuz. Amaç verilen zamanda kaya parçalarını iterek veya düşürerek elmasları toplayıp gizlenmiş çıkışa ulaşmak. Çıkışa doğru emin adımlarla yol alırken düşen bir kayanın altında kalmak, kayaların arasında hapsolmak, gerekli bir elması kaybetmek, çıkışın daha önce ittiğiniz kayalar yüzünden ulaşılamaz hale gelmesi ve sürenin bitmesi de mümkün.

Oyuna başladığınızda sol üst tarafta mağarada kaç elmasın olduğu ve çıkışa ulaşmak için kaç elmas toplamanız gerektiği yazıyor. Gizli olanları bulmak size kalmış. ☺ Gizli çıkışın ortaya çıkması için 150'den geriye sayan sayaçtan önce yeterli sayıda elmas toplamalısınız. Yeterli sayıda topladıysanız ve hala vaktiniz varsa kalan elmasları da almayı ihmal etmeyin.

Ayrıca bölümden bölüme mağaralarda karşılaştığınız şeyler de değişiyor. Öldürücü kelebekler, ateş böcekleri, yıkılmaz titanyum duvarlar, yer çekimini değiştiren teleporterlar gibi.

Oyunda 16 değişik mağara ve her mağaranın 5 zorluk seviyesi mevcut. 500 puana ulaştığınızda bonus olarak bir can alıyorsunuz. Her dört mağaradan sonra "intermission" denilen bonus bölümlere geçiyorsunuz eğer burada da başarılı olursanız yine bir can kazanıyorsunuz ama bu bölümleri çözüp geçmek bir hayli zor.

Künye:

Boulder Dash I
Yıl:1984
Programlayan: Peter Liepa
Geliştiren: Micro Fun
Yayınlayan: First Star Software

Boulder Dash II
Yıl:1984
Programlayan: Peter Liepa
Geliştiren: Megabyte
Yayınlayan: First Star Software

Boulder Dash III
Yıl:1986
Yayınlayan: First Star Software

Boulder Dash Construction Kit
Yıl:1986
Programlayan: Jeff Schneider
Geliştiren: First Star Software
Yayınlayan: First Star Software

Şeklinde liste uzuyor.

BDDB: the Boulder Dash® database http://boulder.cl4.org/ adresinde Boulder Dash' in ilgili 268 oyununun, künye bilgilerini ve resimlerini bulabilirsiniz.

Aslında bu oyunları indirmek yasal olmasa da http://www.c64.com/, ftp://arnold.c64.org, http://www.firefox.ch, http://www.c64unlimited.net/, http://www.lemon64.com/ vs... adreslerden bazı oyun ve ses efektlerine ulaşmanız mümkün.

Commodore tutkunları için Boulder Dash vazgeçilmez klasiklerden biri. Ben her oynadığımda aynı tadı alıyorum. Umarım sizler de Rockford'un mağaralarındaki elmasları hala topluyorsunuzdur.


Rockford™

PROGRAM DÖKÜMLERİ

PROGRAM ADI: CIZGI KURSOR 0801 0860

```
0801: 0B 08 D4 07 9E 32 30 36  - 43A8
0809: 31 00 00 00 78 A9 34 A2  - 57E0
0811: 08 8D 14 03 8E 15 03 A2  - 472E
0819: 00 8A 9D C0 03 E8 E0 40  - 829E
0821: D0 F8 A9 FF 8D D5 03 A9  - A394
0829: 0F 8D F8 07 A9 01 8D 15  - 5291
0831: D0 58 60 EA A5 D3 C9 1D  - 90D4
0839: 90 03 A2 01 2C A2 00 8E  - 55EC
0841: 10 D0 0A 0A 0A 18 69 18  - 2D8F
0849: 8D 00 D0 A5 D6 0A 0A 0A  - 4E02
0851: 18 69 32 8D 01 D0 EE 27  - 6B24
0859: D0 20 EA FF 4C 61 EA 27  - 87C9
```

PROGRAM ADI: KARAKTER SET 2 0801 086C

```
0801: 0B 08 D4 07 9E 32 30 36  - 43A8
0809: 31 00 00 00 78 A5 01 29  - 333E
0811: FB 85 01 A9 D0 A2 30 A0  - 8AF0
0819: 00 84 FB 85 FC 84 FD 86  - A869
0821: FE A2 10 B1 FB 91 FD C8  - BC58
0829: D0 F9 E6 FC E6 FE CA D0  - E25B
0831: F2 A5 01 09 04 85 01 58  - 4593
0839: A9 1C 8D 18 D0 A9 30 A0  - 7ADD
0841: 00 84 FB 85 FC A0 00 B1  - 8DD9
0849: FB 48 C8 C0 08 D0 F8 A0  - A95B
0851: 00 68 91 FB C8 C0 08 D0  - 965E
0859: F8 A5 FB 18 69 08 85 FB  - 9301
0861: 90 02 E6 FC A5 FC C9 40  - A244
0869: D0 DB 60 00 00 00 00 00  - 26B1
```

PROGRAM ADI: HADES BASIC C000 C1EE

(NOT : Programı yükleyin, NEW komutu verin, sonra SYS 49152 komutu ile yeni komutları aktif hale getirin.)

```
C000: A9 0B A2 C0 8D 08 03 8E  - 63C8
C008: 09 03 60 A5 7A A6 7B 85  - 7541
C010: FB 86 FC A9 65 A2 C0 85  - A572
C018: FD 86 FE A2 00 A0 00 20  - 626B
C020: 73 00 85 02 B1 FD C5 02  - 6DD3
C028: D0 13 C8 98 DD 5B C0 D0  - A903
C030: EE 8A 0A AA BD 9C C0 48  - 8AFB
C038: BD 9B C0 48 60 BD 5B C0  - 933A
C040: 18 65 FD 85 FD 90 02 E6  - 97AA
C048: FC E8 EC 5A C0 D0 D5 A5  - BECE
C050: FB A6 FC 85 7A 86 7B 4C  - 8A0D
C058: E4 A7 0A 06 04 06 04 06  - 207D
C060: 06 06 05 06 05 53 50 52  - 2E49
C068: 49 54 45 53 50 52 B9 53  - 60FB
C070: 50 52 50 4E 54 53 50 52  - 514D
C078: BD 53 50 52 4D 55 4C 53  - 5819
C080: 50 52 4D 43 4C 53 50 52  - 4ED1
C088: 43 4F 4C 53 50 52 49 B0  - 6456
C090: 53 50 52 41 4C 4C 43 55  - 4CB0
C098: 52 53 B0 AE C0 C2 C0 F8  - BA53
C0A0: C0 08 C1 30 C1 44 C1 53  - 775C
C0A8: C1 63 C1 77 C1 87 C1 20  - 8427
C0B0: AB C1 B0 06 2D 15 D0 4C  - 657C
C0B8: BD C0 0D 15 D0 8D 15 D0  - 7F37
C0C0: 4C AE A7 20 C8 C1 20 FD  - 9823
C0C8: AE 20 EB B7 8E DA C1 AE  - AF7D
C0D0: DB C1 A5 15 F0 09 BD DC  - 9D68
C0D8: C1 0D 10 D0 4C E5 C0 BD  - 9B64
C0E0: E4 C1 2D 10 D0 8D 10 D0  - 8363
C0E8: 8A 0A AA A5 14 9D 00 D0  - 74EC
C0F0: AD DA C1 9D 01 D0 4C AE  - 9140
C0F8: A7 20 C8 C1 20 D3 C1 8A  - 9662
C100: AE DB C1 9D F8 07 4C AE  - 9240
C108: A7 20 AB C1 B0 06 2D 1D  - 56C3
C110: D0 4C 17 C1 0D 1D D0 8D  - 72F5
C118: 1D D0 20 D3 C1 20 B1 C1  - 90F7
C120: B0 06 2D 17 D0 4C 2B C1  - 6A52
C128: 0D 17 D0 8D 17 D0 4C AE  - 7A5A
C130: A7 20 AB C1 B0 06 2D 1C  - 5688
C138: D0 4C 3F C1 0D 1C D0 8D  - 7668
C140: 1C D0 4C AE A7 20 9B B7  - 87AF
C148: 8E 25 D0 20 D3 C1 8E 26  - 776F
C150: D0 4C AE A7 20 C8 C1 20  - 7D12
C158: D3 C1 8A AE DB C1 9D 27  - 94FE
C160: D0 4C AE A7 20 AB C1 B0  - 9A11
C168: 06 2D 1B D0 4C 72 C1 0D  - 5950
C170: 1B D0 8D 1B D0 4C AE A7  - 877C
C178: 20 9B B7 20 B1 C1 B0 03  - 6FE7
C180: A9 00 2C A9 FF 4C BD C0  - 96B4
C188: 20 9B B7 E0 19 B0 19 8E  - 78A4
C190: EC C1 20 D3 C1 E0 28 B0  - 9F7D
C198: 0F 8E ED C1 AE EC C1 AC  - B46C
C1A0: ED C1 20 0C E5 4C AE A7  - 8B5E
C1A8: 4C 48 B2 20 C8 C1 20 D3  - 87E0
C1B0: C1 E0 02 B0 F3 E0 01 F0  - A6D5
C1B8: 07 AE DB C1 BD E4 C1 60  - A3CB
C1C0: AE DB C1 BD DC C1 38 60  - 9830
C1C8: 20 9B B7 E0 08 B0 D9 8E  - 9555
C1D0: DB C1 60 20 FD AE 20 9E  - 8A71
C1D8: B7 60 00 00 01 02 04 08  - 162C
C1E0: 10 20 40 80 FE FD FB F7  - BC23
C1E8: EF DF BF 7F 00 00 00 00  - 3FF8
```

SON SAYFA

LAMERS TURİZM 2004 DÜNYA TURU

ÖZELLİKLE SCENE AĞIRLIKLI TURLARIYLA TANINAN, TURİZM SEKTÖRÜNÜN LİDERİ LAMERS TURİZM TARAFINDAN DÜZENLENEN DÜNYA TURUNA TÜM LAMER'LERİ DAVET EDİYORUZ. DÜNYACA ÜNLÜ LAMERLERLE TANIŞMAK VE GERÇEK BİR LAMER OLMAK İSTİYORSANIZ BU FIRSATI KAÇIRMAYIN.

KONTENJANIMIZ SINIRLI OLUP TURA KATILMAK İÇİN ACELE EDİN. **LAMERS TURİZM** ACENTALARINDAN AYRINTILI BİLGİ ALABİLİRSİNİZ.

TUR PROGRAMI : GECE YAPILACAK OLAN TANIŞMA PARTİSİNDEN SONRA, SABAH LUX OTOBÜSLERLE YOLA ÇIKILACAKTIR. TUR PROGRAMINA GÖRE UĞRANILACAK OLAN SCENE PARTY'LERDE SÜRPRİZ YARIŞMALAR VE HEDİYELER SİZLERİ BEKLİYOR....

LAMERS TURİZM

ASLA OLMAYACAK ŞEYLER LİSTESİ

1 - PC'LER ARTIK C64 COMPATİBLE OLARAK ÜRETİLSİN.
2 - MİCROSOFT İFLAS ETSİN. BİLL GATES VE EKİBİ C64 İÇİN ÇALIŞSIN.
3 - HARDWARE VE SOFTWARE FİRMALARI C64 İÇİN ÜRÜNLER GELİŞTİRSİN.

OLMAYACAK KİŞİSEL İSTEKLER

HANGİ DİLDE OLURSA OLSUN COMMODORE 64 İLE İLGİLİ OLARAK YAYINLANMIŞ BÜTÜN KİTAPLARA VE DERGİLERE SAHİP OLMAK.
20 SENE ÖNCEKİ SAÇLARIMA KAVUŞMAK.
BMW 540 SAHİBİ OLMAK.
.....
.....