Plazma 4

Plazma 4

Yayımlanma 2008

İçindekiler

Editörden

Emirhan 'Ragnor' Bayyurt

Bilgem 'Nightlord' Çakır

Ragnor: Selam Plazma okurları...Ben Ragnor, bu sayıdan itibaren editörlük görevini Spritus'tan devralmış bulunmaktayım. Geçen sayıda girişi böyle yapmıştım. Şimdi ise ben Ragnor, bu sayıdan itibaren editörlük görevini Nightlord'a devretmiş bulunmaktayım. Fazla dayanamadım anlaşılan. Şaka bir yana, öncelikle geçen sayıdan bu yana geçen uzun zaman yüzünden hepinizden özür dilerim. Acemiliğim yüzünden yaşadığımız problemlerle bir çıkmaza girmiştik ki Nightlord cesurca bir hamle ile editörlüğü üstlenmeyi teklif etti. Açıkçası kendisinin, benden çok daha iyi iş çıkaracağına eminim. Huzurunuzda kendisine tekrar teşekkür ediyorum. Bu sayıdaki en radikal değişiklik dergi arayüzünün web'e ve pdf'e taşınması oldu. Açıkçası, exe arayüz'ün geliştirilmesi ile ilgili olabilecek problemleri çözen oldukça cesur ve güzel bir hamle. Beni üzen tek şey ise dergi arayüzü için müzik bestelemiş ya da arayüzü çizmiş olan Hyper, Punky ve LordHW arkadaşlarımızın emeklerinin karşılığını verememiş olmam ama inanıyorum ki onlar da zamanı gelince değerlendirilecektir (Ntl: Hyper'ın muhteşem kapak grafiği, Hydrogen'in aynı güzellikteki kompozisyonu ile pdf versiyonumuzun kapağı olarak değerlendirildi bile ;)). Umarım beni affedersiniz :(.

Neyse benden bu kadar, elimden geleni yapmaya çalıştım, az da olsa bişeyler yaptım ve bayrağı devrediyorum. Hepinize mutlu ve üretkenlik dolu bir yıl diliyorum.

Şekil 1.

Nightlord: Soğuk bir Şubat gecesi, derginin son hazırlıklarını tamamlamaktayım. Birkaç saat sonra, dergiyi web sitesine yükleyip, ardından forum forum dolaşıp, derginin çıktığını duyuruyor olacağım. Böylece son üç haftadır evdeki zamanımın büyük bölümünü alan bu proje, bir sayısını daha geride bırakmış olacak. Şu an buna bu kadar yaklaşmış olmak ilginç bir duygu. Bir hayli emek harcayıp, şu anki haline getirdiğimiz derginin bu sayısında, halen “keşke daha çok vakit olsaydı; şurasını daha geliştirebilirdik” dediğimiz pekçok yer var. Fakat Plazma ekibinin şu anki öncelikli hedefi, dergiyi düzenli bir takvime oturtmak. Her ne kadar bu dergiyi her yönden geliştirmek istiyorsak da, öncelikli kaygımız sayıların bundan böyle “hiç” geç kalmaması. Dergimiz bundan sonra üç ayda bir çıkıyor olacak. Ocak, Nisan, Temmuz ve Ekim aylarının 15’inde çıkıyor olacağız.

Bu hedefe rağmen, şu an okumakta olduğunuz sayı, malesef bir hafta gecikti ve 28 Ocak’ta çıkması hedeflenirken, 4 Şubat’ta çıkıyor. Bu gecikmenin, okuyucuların ihtimalen umrunda olmayan çeşitli sebepleri var elbette. Bu sebeplerin en önemlisi, az sonra dergiyi okumaya başladığınızda farkedeceğiniz üzere, dergide normalin üstünde miktarda içerik olması. Bunun sonucu olarak, bütün yazarlar yazılarını vaktinde teslim etmiş olmasına karşın, yazıların düzenlenip derginin bir araya getirilmesi, ilk planladığımızın üstünde zaman aldı. Sonraki sayılarda bu problemle karşılaşmamak ve asla gecikmemek için önlemler alıyor olacağız.

Bu içerik bolluğu, gecikme gibi bir negatifliğe dolaylı olarak yol açmış olsa da, elinize ulaşan bu derginin değerini de çok artırmış olacağını düşünüyoruz. Plazma dergisi, pekçok iyi yazardan, çok fazla miktarda bilgi içeren, çok kaliteli bazı yazıları bu sayıda sizlere ulaştırıyor. Yazılardan ve yazarlarımızdan bahsetmeden önce, bu sayı ile beraber Plazma’da olan iki önemli değişiklikten bahsetmek istiyorum.

İlk farkedeceğiniz değişiklik, Ragnor'un da az önce değindiği gibi, derginin ulaşılabildiği formatlarda oldu. Yayın hayatına “Demoscene” geleneklerine uygun bir “Disk Dergisi” (yani exe) olarak başlayan Plazma, bu sayıdan itibaren, “daha geniş kitlelerce daha kolay erişilebilir olmak” hedefi doğrultusunda, web sayfası ve pdf formatlarında karşınıza çıkıyor. Bu sayede, dergimizde yayınlanan yazıların tamamına, ister web sayfası olarak ulaşabilecek , ister pdf versiyonunun yazıcı çıktısını alıp, basılı bir dergi gibi kullanabilecek veya arşivleyebileceksiniz. Bir diğer önemli kazanç da, yazıların web arama motoru robotları tarafından taranıp, “web de aranabilir” hale gelmesi olacak. Bu format değişikliği esnasında, geçici olarak “Disk Dergisi” formatına ara veriyor olacağız. Yeni bir disk dergisi arayüzü bir yandan geliştiriliyor ve o arayüz yeterli olgunluğa geldiğinde dergi, web ve pdf formatları ile beraber, tekrar exe formatında da çıkıyor olacak. Kesin olmamakla birlikte, yeni exe arayüzünün, derginin altıncı sayısı ile geri döneceğini planlıyoruz. Bu noktaya kadar geçen süre zarfında, çıkmış olan geçmiş sayıların da (yani dördüncü ve beşinci sayılar) exe versiyonları o zaman çıkartılacak. Benzer şekilde, sadece exe formatında çıkmış olan ilk üç sayının da pdf ve web arayüzlü versiyonlarını, önümüzdeki zamanda çıkararak, onları da daha “erişilebilir” hale getirmeyi hedefliyoruz.

İkinci değişiklik de derginin içerik odağının genişlemesi. Daha önceki sayılarda, temel olarak “Demoscene” çerçevesinde, ağırlıklı olarak PC ve C64 platformlarına dair içeriğe sahip olan dergimiz, kapsama alanını bundan böyle önemli ölçüde genişletiyor olacak. Başka bir deyişle, Plazma artık bir “Demoscene” dergisi olmanın ötesinde, bir “Amatör Bilgisayar Kültürü” dergisi olmayı hedefliyor olacak. Bunun sonucu olarak dergimizde bundan böyle, her türlü donanımsal ve sanal platform üzerinde yapılan,

amatör yazılım ve donanım geliştirme veya grafik, müzik gibi sanatsal ve yaratıcı çalışmalar üretme üzerine yazılar görüyor olacaksınız. Eğitici kurs nitelikli yazılar, çeşitli yazılım ve donanım araçlarının tanıtım ve kurulum yardımı yazıları, yapılan etkinliklerin raporları veya çeşitli amatör eserlerin inceleme ve yorumlamaları gibi pekçok türde makalelerimiz olacak. Amacımız, herhangi bir platformda, "amatör" ve "bilgisayar ile ilgili" birşeyler üreten veya üretmek isteyen insanlara yardımcı olmak, karşılaşabilecekleri teknik problemlerle ilgili bir bilgi ve destek ağı oluşturmak. Bir diğer amacımız da, bilgisayar ile ilgili herhangi bir konuda birşeyleri "kurcalayan", "problem çözen" insanların kendileri gibi başkalarını bulup yardımlaşmalarına aracı olmak. Başka bir deyişle Plazma, bilgisayarı ile değişik teknik problemleri çözen insanların arasında bir "katalizör" rolü oynamaya çalışacak.

Şekil 2.

Bu yeni kapsam genişlemesinin sonuçlarını, bu sayımızda hemen görmeye başlayacaksınız. Daha önceki sayılardan farklı olarak, bir anda en fazla kapsam patlaması olan platform bu sayıda "Amiga" oldu. Bu sayımızda, assembly programlamadan, sistem kurulumuna ve hardware rom hacklerine kadar birbirinden ilginç konuları kapsayan üç uzun ve kaliteli Amiga yazısı bulacaksınız. Bu sayıda daha çok klasik Amigalar kapsama alanımızda olsa da, ilerleyen sayılarda daha modern sistemler ve özellikle de Amiga OS ve MorphOS gibi yeni nesil Amiga işletim sistemleri hakkında yazılar da içeriyor olacağız.

Yine genişleyen kapsamımızın sonucu olarak, "üç boyutlu grafik" ve "modern müzik sequencerlar" ile ilgili kurs yazı dizileri de başlattık. Bu derin ve önemli alanlarda ilerlemek isteyen okuyucularımıza Türkçe kaynak yaratmaya başlıyor olduğumuz için çok mutluyuz. Bu yazılara, daha klasik platformlarda müzik ve grafik çalışmaları yapmak ile ilgili, usta "oldskool" grafiker ve müzisyenlerin hazırladığı yazılar da eşlik ediyor olacak. Bu usta yazarların yazılarında, platformdan bağımsız olan pekçok sanatsal ve teknik teorik bilgiyi de buluyor olacaksınız.

Web teknolojileri, PC platformları ve genel yazılım mühendisliği odaklı yazılarımız da bu sayı ile beraber artarak geliyor olacak. Yeni başlayan PHP kursu ve yazılım geliştirmede test tekniklerini tanıtan bir makale bu sayıda hizmetinizde. İlerleyen sayılarda programlama dilleri, çeşitli sık kullanılan kütüphanelerin (C++ STL, Boost, DirectX, OpenGL, SDL vs.) tanıtımları, ve daha ileri bazı yazılım mühendisliği konularını (Refactoring, Tasarım Desenleri vs.) sunmaya devam ediyor olacağız. Tabii bu yazılarımız, yine daha klasik platformlara yönelik geliştirme teknikleri ve türlü kurslarla geniş bir yelpazeyi paylaşacaklar.

Ülkemizde çok tanınmayan bazı klasik platformları da, daha derinlemesine tanıma imkanı bulacaksınız. Bu sayıdan itibaren Amstrad ve Spectrum köşelerimiz, bu daha az yayılmış ama oldukça güçlü amatör kullanıcı topluluklarına sahip ilginç makineleri tanımanız için kapıyı aralıyorlar.

Bu sayının ağır toplarından biri de ülkemizdeki "Bilgisayar Scene" tarihini, dilden dile anlatılan folklorik bir olgu olmaktan kurtarıp yazılı olarak kayıt altına geçirmeye çalışan "Türk Scene Tarihi" yazı dizisinin ilk parçası. Bu süper çalışmayı yapan değerli yazarımız Vigo'nun da belirttiği gibi, bütün eski Scene üyelerini, bu tarihi kaydetme konusunda, bize yardım etmeye çağırıyoruz. Her türlü ekleme ve düzeltmeler için bize ulaşın. Ayrıca yeni nesillerin Türkiye'deki bilgisayar trendlerini daha iyi algılamalarına bu yazının aracı olacağını umuyoruz. Bütün bu "ulvi" amaçların yanında, Vigo'nun eğlenceli ve samimi anlatımını okurken çok eğleneceğinizi de tahmin ediyoruz.

Bu sayımız ayrıca çeşitli güncel olaylarla ilgili yazıları da size ulaştırıyor. Geçen yıl yapılan Nightshift 07 ve 7d7 partilerinin raporları, yeni kişisel konsol GP2X hakkında bir inceleme ve demoscene ile ilgili ilk izlenimlerden tutun da, partilerde yayınlanan bazı amatör ürünlerin inceleme ve yorumlarına kadar pekçok yazıyı beğeninize sunuyoruz.

Umarız bütün bu yenilikler, okuyucularımızın her yeni sayısını heyecanla bekleyip arkadaşlarıyla paylaştıkları, okuması zevkli ve eğlenceli bir Plazma yaratmamızı sağlar. Ülkemizde teşvik edip desteklemeye çalıştığımız "Amatör Bilgisayar Kültürü" projelerinin, hem sayıca artmasına ve başarılı olmasına yardımcı olmak, hem de bunu yaparken kendisi bir "Amatör Bilgisayar Kültürü Projesi" olarak örnek teşkil etmek en büyük dileğimiz.

Dergi ile ilgili istek ve eleştirilerinizi her zaman bize ulaştırabilirsiniz. Plazma'nın 5. Sayısıyla 15 Nisan 2008 tarihinde tekrar karşınıza gelene kadar, hoşçakalın.

nightlord (at) nightnetwork (nokta) org

Şekil 3.

Nightshift 07 Parti Raporu

Ege Şafak 'Punky' Erkul

2 Mart cuma - İstanbul'a Yolculuk

Parti günü gelip çatmıştı işte. Sabah 4'te kalkıp genel bir hazırlık yaptım. Son kontrolleri de yaptıktan sonra saat 7 civarı evden çıkıp Mgzzy'nin evine uğradım. Zile basmadığım halde nasıl bir rastlantıdır ki tam kapısına vardığımda kapıyı açtı :) Yanında değil çanta ufak bir torba bile olmadan evden ayrıldı. Benim yanımda ise içinde bir ton şey olan bir sırt çantası vardı. Otobüs terminaline gittik, otobüse bindiğimizde saat 9 civarıydı. Sonunda İzmir'den ayrılıyorduk.

Şekil 1.

Yol boyunca uyumaya çalıştık, parti mekanında uyanık olmamız lazımdı. Tüm çabalarımıza rağmen uyuyamadık ve yorgunluğumuzu gideremedik. Vapur seyahatinin geldiği sırada yavaş yavaş uyumaya başlayan Mgzzy'i uyandırdım. "Uyanık ol vapurda gezeriz" gibisinden birşeyler dedim, ardından farketmeden ben uyudum :) Mgzzy beni uyandırdığında otobüsümüz vapurdaydı. Kapılar açılınca ilk işimiz tuvaleti ziyaret etmek oldu. Dalgalardan dolayı, sallana sallana işemek zor bir işti, bunun da üstesinden geldik…

Akşam 6-7 civarı yolculuk bitti, İstanbul'a ayak bastık! Mgzzy, İstanbul'daki arkadaşı Beelzebub ile telefonda görüştü, Taksim'de buluşmayı planlıyorduk fakat olmadı. Beelzebub doğruca parti mekanına gitti, biz de kısa süreli özgürlüğümüzden yararlanıp Taksim'e uğradık. Birşeyler yemek için önce Mc Donalds'a uğradık, buradaki hayvani fiyatları gördükten sonra karşı caddedeki Simit Sarayı'na gittik. Birşeyler yedikten sonra parti için hazırdık! Ama bir saniye, gideceğimiz yer neredeydi? Taksim Meydanı'nda etraftaki taksicilere yolu sorduk sürekli. Hiçbirisi de Bilgi Üniversitesi'nin Kuştepe Kampüsü'nü bilmiyordu. Ben sürekli Mgzzy'e "Vigo'nun anlattığı yolu izleyelim" diyordum, o da taksiyle gitmeyi savunuyordu.

Sonunda onu da ikna ettim, Vigo'nun anlattığı yolu denedik. Mecidiyeköy'e gitmek için metro aradık etraflarda. Gözümüzün önündeki metro istasyonunu uzaklardaki bir gazete bayisinden öğrendik, ne kadar dikkatsizmişiz meğersek! Gazeteci elemanın dediğine göre Mecidiyeköy yazarmış metroda. Biz de girdik metroya ve gideceğimiz yönü aradık. Etrafa çok yabancıydık, sanki başka bir ülkeden gelmiştik. Eminim partideki tek yabancı ülkeli arkadaş Xc8 bile İstanbul'u bizden daha iyi biliyordur :) Metroya bindik, etrafta durak çizelgesini aradık. Uzakta bir durak çizelgesi vardı ama Mecidiyeköy yazmıyordu. Mgzzy'e "olm sanki yanlış şeye bindik gibi" dedim. O da baktı çizelgeye, biraz daha dikkatli baktı ve 32 puntoluk -sanırım- "Birinci Levent" yazısının altındaki 10 puntoluk "Mecidiyeköy" yazısını gördü. "Has***ktir" diyerek birbirimize baktık, çünkü bir durak geçmiştik Mecidiyeköy'ü :) Hemen karşı yöndeki metroya atlayıp Mecidiyeköy'e vardık. Bir otobüs durağındaki Milli Piyango'cu amcaya yolu sorduk ve biraz muhabbet ettik. O da yolu bilmiyordu, bize Bornova'da turşucu bir arkadaşı olduğunu ve tanıyıp tanımadığımızı sordu :) Tanırız tabii Bornova'da kaç tane turşucu olabilir ki :P … Neyse, otobüse binme fikrimiz de boşa çıktı. En sonunda yakınlarda bir yerlerde taksiye atladık veee parti mekanına vardık!

Nightshift'e Hoşgeldik

Üniversiteye adım attıktan sonra Nightshift posterinin olduğu kapıya yöneldik. Posterleri takip ederek parti mekanı olan study hall'e vardık. Ortama ilk ayak basışımda oldukça heyecanlıydım. Yeni bir ortama girişin heyecanı üzerimden geçtikten sonra kendime geldim, ilk olarak bizi karşılayan Decipher'la tanıştım. Sonrasında da Ragnor, Stranger, Vigo ve ortamdaki diğer kişilerle tanıştım.

Biz geldikten 15 dakika kadar sonra Beelzebub geldi. Onunla da tanışmış oldum. Hazır biz birlikte iken fotoğraf makinesi olanlar birden üstümüze gelip fotoğrafımızı çektiler.

Şekil 2.

Biraz muhabetten sonra üst kata çıkıp adımıza ayrılmış bilgisayarların başına geçtik. Ortamda sahiden de 100mbit'lik bağlantı hızı vardı. Evde 256k ile indirmek için dakikalarca beklediğim dosyalar göz açıp kapayıncaya kadar iniyordu, bu çok güzel bir his… Biraz zaman geçtikten sonra yarışacak ürünlerimizin olduğu cd'yi cdrom sürücüye bağlamayı denedik ama bilgisayarların cdrom sürücüleri makineye bağlı değildi (??!) Biz de cd'yi Vigo'ya verdik ve o da dosyalarımızı kendi makinesinden bizim makinemize aktardı.

Ed: İşte Nightshift partilerinin arkasındaki itici güç: Vigo ve Hyper

Şekil 3.

Muhabbet, bilgisayar, iş-güç derken birisi mekana Wii getirdi!! Aşağı katta Skate ve Ragnor adeta hipnoz edilmiş bir biçimde Wii'de karşılıklı tenis oynuyorlardı. Ben de aşağıya inip Wii'yi kullanma fırsatı yakaladım. Doğrusu Wii gerçekten çok eğlenceli bir araç, keşke fiyatı da kendisi kadar güzel olsa… Wii oynamayı bıraktıktan sonra yukarıya çıkıp işlerime devam ettim.

3 Mart Cumartesi

Gecenin ilerleyen dakikalarında insanlar yavaş yavaş uyumaya başladılar. Başı boş Wii'yi gören Mgzzy hazır fırsatımız varken Wii oynamayı önerdi. Ben de uykulu bir şekilde elimdeki Wii-mote'u sağa sola sallamaya başladım ve sürekli yenildim :) Wii'yi kapattıktan sonra saat 4 civarı masalarımızın dibine çöktük ve uyuduk.

Sabah saat 6:30-7:00 civarı kollarım uyuşmuş bir şekilde uyandım. Mgzzy'i uyandırmaya çalıştım, uyanmadı. Ben de tuvalete kısa süreli bir yolculuk yaptım. Bu kısımdan sonrasını pek hatırlamıyorum, uykuluydum. Bir süre sonra insanlar yavaş yavaş uyanmaya başladılar, anons yapıldı ve herkes dışarıya çıkıp kaydını yaptırıp tekrar parti mekanına döndü. Kahvaltı ikramı vardı; açma, poğaça vs. karnımı doyurdum, bir süre sonra Hydrogen'in 3dmax ve 3d ile ilgili semineri başladı. 3d'den, modellemeden anlamayan biri olduğumdan pek fazla ilgimi çekmedi, ama oldukça güzel geçti. Saat 15:00 civarı 8 renk piksel grafik yarışması vardı. Tabii ki büyük üstad turbo kazandı. Ardından merakla beklediğim iki seminerden ilki, vst tarafından sunulan "Windows'u unut, Linux kur" semineri vardı. seminerde anlatılanlar benim zaten bildiğim şeylerdi, bence çok da faydalı bir seminer değildi.

Ed: İşte Türk Demoscene'in en yakışıklı ve endamlı grubu Glance :) Bir üyeleri katılamadı partiye maalesef

Şekil 4.

Ed: Türkiye'nin en iyi aktif pc demo coder'ı Anesthetic/Resident seminerini verirken

Şekil 5.

Akşam 18:00 civarında merakla beklediğim ikinci seminer, Anesthetic tarafından sunulan "ilk intro'm" semineri vardı. Anes bizler için canlı olarak bir intro kodladı. Seminer benim için çok öğretici oldu, eve gittiğimde pixeltoaster ile birşeyler kodlamaya başlamıştım :) Sonraları Commodore ile ilgili bir seminer olmuştu, üst katta işlerim olduğu için bu seminere katılamadım. Vigo'nun semineri ise insanların yorgunluğu ve saatin geç ol-

ması nedeniyle iptal edildi. Gecenin ilerleyen dakikalarında ise Stumn 201 konseri vardı. Çok güzel bir konser veriyorlardı, ortamdaki müthiş sese rağmen deliler gibi uykum geliyordu. Belki uykum geçer diye Endo'nun oyun yazmasına yardım eden Mgzzy ve Beelzebub'ın yanına uğradım. Bu iki arkadaşım Endo'ya sürekli güzel fikirler veriyorlardı, ben de uykulu bir şekilde Endo'nun oyun yazışını izliyordum. Logo çizimi için Turbo geldi ve ilk kez Turbo'yu bir iş başındayken izleme fırsatı yakaladım. Hiç tereddüt etmeden "smooth" bir şekilde logo çizişini görünce hayranlıklar içinde kaldım. Ortamdaki müthiş ses ve Turbo'ya rağmen uykuma dayanamadım ve masamın başına geçtim, klavyeyi ileriye ittim ve uyuklamaya başladım. Uyandıktan sonra öğrendim, yanlışlıkla klavyenin üzerine başımı yaslayarak uyumuşum ve bilgisayarın sürekli biplemesine neden olmuşum :)

Şekil 6.

4 Mart Pazar

Bugün compo günüydü. Maalesef katılım çok azdı, bazı compolar iptal edilmek zorunda kaldı. Ama buna rağmen music compo'ya inanılmaz bir ilgi vardı, yarışan o kadar çok müzik vardı ki iptal olan compoların sürelerini biraz olsun doldurmayı başardı. Ben music compo'ya ve gp2x skin compo'ya katıldım. Music compo'da 6. oldum, fakat benden başka katılan olmadığı için gp2x skin compo iptal edildi. Compo'lar sırasında insanlar neredeyse zorla alkışlarken gp2x executable compo başladı, Bronx'un "check this out" isimli demosu dev ekranda gözüktü ve insanlar birdenbire çılgına döndü! Demo bitince sonunda o özlenilen büyük alkışlardan biri daha salonda yankılandı. Aklıma kazınan diğer ürünlerden biri de Rephisto'nun "i don't need a head" çalışmasıydı.

Şekil 7.

Sonunda compo'lar bitti, ödül törenine geçildi. Turbo'nun neredeyse her compo'da birincilik kazanması nedeniyle ödül için birçok defa çağırılması sonucu çok eğlenceli dakikalar yaşandı. Wisdom/Crescent music compo'da, Turbo/bronx live battle'da, pixel graphic compo'da ve on the fly compo'da, Endo/glance game compo'da, Bronx grubu ise gp2x executable ve combined demo compo'da birinci oldular. Katılım azlığı nedeniyle bazı compo'ların iptal edilmesi sonucu parti beklenilenden oldukça erken bir saatte bitti. Ve herkes evlerine dağıldı…

Demoscene ile Tanışma

Atilla Filiz

Ne olduğunu daha önceden biliyor olmama karşın, demoscene ile adam akıllı tanışmam Nightshift 2007'de oldu. Daha önceden demo izlediğim olmuştu ama bu şeyleri hazırlayan insanlar hakkında herhangi bir fikrim yoktu.

Burada Nightshift'ten demolardan vs. bahsetmeyeceğim, eminim bu konuda bolca malümata kolayca ulaşılabiliyor. Ben burada bu işlerle uğraşan cemaatten, scene denen şeyin kendisinden bahsedeceğim.

Şekil 1.

Nightshift'e(bundan sonra NS07 diyeceğim) giderken orada beni çeken bir numaralı şey demo falan değildi. Güzel demolar izlemeyi de umuyordum ama beni esas oraya çeken şey, eski bilgisayarlara, özellikle 8-bit olanlara, özellikle de Commodore 64'e olan özel ilgimdi.

Ortama ilk girdiğimde kendimi çok yabancı hissettim. Zaten etkiniliğe bir gün geç gitmiştim, oradakilerin de bir çoğu bilgisayarlarına gömülmüş, bir şeylerle uğraşıyorlardı. Kod yazan, kulaklığını takıp müzik besteleyen ve değişik yöntemlerle resim çizen bir sürü insan vardı. Demoscene'in özünü de bu oluştuyor: bir şeyler üretmek. NS07'de, diğer benzer etkinliklerde olduğu gibi, çeşitli yarışmalar vardı ve orada gördüklerimin çoğu bu yarışmalar için bir şeyler hazırlıyorlardı. Bazılarının yanına gidip "abi n'apıyorsun", "usta bunu neyle yapıyorsun" ya da "hoca bunu nereden buldun" gibi sorular sordum. Yaptıkları şeylerle ilgilenildiğini görenler büyük bir zevkle cevap verdiler. Oraya çok hazırlıksız gitmiştim(bilgisayarım yanımda değildi ve bir şeyler yapabilmem için GERÇEKTEN ona ihtiyacım vardı) ama oradakiler(özellikle evsahibi vigo) bir şeyler yapmam için sürekli gaz verdiler. Uğraşmam, bir şeyler ortaya çıkarmam için bana dil döktüler.

Şekil 2.

Daha sonra bu elemanların çoğunun takıldığı foruma üye oldum. Orada konuşulanlar da genel olarak bir noktada toplanıyor: bir şeyler üretmek. Kimsenin eğlence dışında bir kazancının olmadığı bir hobi için ne kadar ciddi tartışmaların olduğu inanılmaz. Bu e-derginin hazırlanmasında da bir çok şey konuşuldu, tartışıldı, arada insanların anlaşamadığı da oldu. Bütün bunlar herhangi birinin çıkarı olduğundan değil, sadece bu güzel eserin, eserimizin, ortaya çıkması ve yapabileceğimizin en iyisi olması içindi.

Şekil 3.

Ben demo olayına biraz uzağım çünkü ilgi alanım doğrudan demo üretmeye değil ama ortaya bir şeyler çıkarmaya. Zaten bu adamlar da tüm hayatı demodan ibaret olan ruh hastaları değil. Bir şeyler üretmekten ve üretildiğini görmekten zevk alan insanlar. NS07'de ve ondan önce bir forumda(GP2X Türkiye) bazı projelerimden bahsetmiştim. O kadar çok insan benimle ilgilendi ki hayret ettim. İlk defa partide tanıştığım insanlar bana "sen

oyun hazırlıyordun, değil mi, o nasıl gidiyor" diye sordu. Demo kavramına ve demo üretimine biraz uzak olsam da, hobi amaçlı, birşeyler üretme amaçlı harika bir yeraltı cemaati bu demo cemaati(şimdi bir haber dergisi çıkar "öldüren demo tarikatı" diye haber yapar).

7d7 Parti Raporu

Emirhan 'Ragnor' Bayyurt

Şekil 1.

Giriş

Partiye gideceğim belli olduktan sonra Ankara tayfası ile iletişime geçmeye çalıştım ama sadece Tesla'yı bulabildim. Kötü haber, biletleri almışlardı. Ama ondan detayları öğrenip, ertesi gün ben de bir tren bileti aldım. Aynı vagon olmayacaktı belki ama, aynı tren ile İstanbul'a gidecektik. Akşam, buluşma yeri olan Dost Kitabevi'nin önünde toplanıp, tren garına gittik. Kadro Anesthetic, Scg, Flexi, Flexi'nin kardeşi ve benden oluşuyordu. Tesla ve arkadaşı Dağhan bizimle garda buluşacaktı. Nitekim öyle de oldu. Bol miktarda geyik yapıldı ve eski yolculuk anıları anlatıldı. Sonunda vakit geldi ve vagonlara geçtik. Tek başına kaldığım için biraz sıkıcı başladı yolculuk ama önemli değildi. Çünkü partide bol bol uykusuz kalacağım için trende tek yapacağım şey uyumak olacaktı. Tabii koltuğumu biraz geriye yaslatabilseydim. Yaslatamadım ve bütün gece türlü abuk pozisyonlara girerek uyumaya çalıştım. Sık sık uyansam da başarısız oldum diyemem.

Şekil 2. Ankara Tren Garı

Gelişme

Sabah olduğunda ufak bir şok yaşadım. Yolculuğumuzun başında hava oldukça sıcaktı ve bu yüzden üzerimde sadece bir t-shirt vardı. Yanımda da sadece yedek bir t-shirt vardı o kadar. Ama İstanbul'da hava kapalıydı, soğuktu ve kimi yerlerde yağmur yağıyordu. Neyse ki biz trenden indikten sonra pek bir yağmurla karşılaşmadık, ama soğuk hava konusunda yapabileceğim bişey yoktu. Sabah ilk iş olarak, Ankara tayfasının her gelişlerinde gittikleri bir kafeye girdik. Ben orada tuvalet sırası beklerken Decipher da mekana damladı ve böylece ekip tam oldu.

Şekil 3. Kadıköy

Kahvaltımızı yaptıktan sonra önce vapurla Beşiktaş'a oradan da otobüsle Boğaziçi Üniversitesi'ne doğru yola çıktık. Üniversiteye geldiğimizde yanlışlıkla (gerçi yol boyunca "abi Güney Kampüsü daha güzelmiş, parti orda olsa keşke" diyen adamlar ne kadar yanlışlık yaptı bu konuda bilemiyorum :)) Güney Kampüsü'ne girdik ve gerçekten çok güzel olduğu için kolay kolay çıkamadık.

Şekil 4. "Yanlışlıkla" gidilen Güney Kampusu

Neyse ki yol sorarak bir iki defa yanlış yola girsek de sonunda kuzey kampüsünü bulduk. Giriş bir zindanı anımsatıyordu. Çe-

şitli Nethack ve Diablo esprileri, ve "Scene mahkumları" temalı birkaç fotoğraf çalışmasından sonra yola devam edip parti mekanına giriş yaptık. Geç kaldık diyorduk ama erken gelenlerdendik ve boş yerlere kurulduk hemen.

Şekil 5. Scene Mahkumları

Gelişme & Tanışma:

Katılım az olsa bile her partide olduğu gibi bu partide de yeni yüzler vardı. Derya Kılıçlı herhalde bu partinin en büyük bombası olmuştu. Eminim ki bundan sonraki partilerin değişmezlerinden olacak kendisi. Bizim için büyük bir kazanç oldu açıkcası. Tabii hepsi bu kadar değil, mesela Tipple ile tanışabildim. İlgili ama nereden başlasam diyen biri, inşallah ortadan kaybolmaz ve bu ilgisini kaybetmez.

Şekil 6. Parti heyecanından gözleri ışıl ışıl olmuş bir grup commodore.gen.tr müdavimi

Maalesef bu partide fazla kişiyle tanışamadım. Bunda en büyük neden, arka tarafta Game Compo'ya oyun yetiştirmeye çalışmam oldu. Bu arada partiye belki de bir demoparty'e bugüne kadar gelen en genç konuk geldi. Skate'in 4 aylık oğlu Kaan ya da nick'i ile Mukmuk parti mekanını şenlendirdi. Ne yazık ki hasta olduğum için gönül rahatlığı ile sevemedim ama görmek bile güzeldi.

Şekil 7. Mekan

Geliştirerek Gelişme:

Sabah partide daha önce Metucon 2006'da tanıştığım Cem Abi'ye (Akyürek) rastlamıştım. Sürpriz Compo'lar açıklanırken kendisi ile muhabbete daldık ve o sırada bana çizdiği grafikleri oyun yapabilecek birini aradığını söyledi. İyi olacak hastanın ayağına gelirmiş ya doktor, hemen atladım tabi. Sonuç eksikleri olsa da (kodlamaya vaktim kalmadı :)) güzel bir oyun oldu. Tabi bu arada birkaç kişiye SDL'in reklamını yaptım. Bu arada klasik oyun turnuvaları yapıldı. Speedball 2'nin Amiga versiyonunu falan emulatörle denemiştim. Ayrıca benzer bir oyundan tecrübeli de sayılırdım (MUDS :)) ama Tron konusunda hiç tecrübem yoktu. Ama iyi bir mücadele çıkarıp Tron'da 2. tura çıkmayı başardım. Sonraki turda usta bir oyuncu çıktı karşıma (sanırım Şükrü idi), ve fazla zorlayamadan elendim. SpeedBall'da ise ilk turda iyi bir mücadele çıkardım ve 2 gol yiyip 2 gol attım ama rakibim (Blackturk idi yanlış hatırlamıyorsam) iki puanlık bir bonus kazanmıştı ve o bonus ile beni yenerek bir sonraki tura çıktı. Bu yenilgiden sonra kodun başına dönmüştüm ki (aslında ilk önce Speedball maçı vardı sonra da Tron :)) scene bilgi yarışması başlayacağı anonsu yapıldı. Yapılan her etkinliğe katılan ben, tabii ki bu yarışmaya da katılmıştım. Sahneye çıkınca durumun cidden kötü olduğunu fark ettim. Bizim takım ben, Anesthetic ve Blackturk'ten oluşuyordu ve hepimizin yüzünde şaşkın bir ifade vardı. Rakip takımda ise Vigo, Wisdom ve yanlış hatırlamıyorsam Nautilius vardı. Bahisler rakip takım kazanacak diyordu. Takım ismi olarakta "pompa"yı seçerek ne kadar iddialı olduklarını ispatlıyorlardı. Ama yarışma başladı ve hatalı sorulara rağmen çok iyi bir yarışma çıkardık ve bütün sorular bittiğinde durum berabere idi. Ara verildi ve yeni sorular hazırlandı. Beklenmeyen gerçekleşti ve biz kazandık.

Şekil 8. Parti müdavimlerinden yükselen değer Demodojo Grubu

Geceleme:

Yarışma sonrasında açlığa artık daha fazla dayanamayıp yemek yemeye çıktık. Yemek güzeldi. Yemekten sonra her zamanki gibi "çay ister misiniz?" diye sordu garson. Başımıza gelecekleri haberimiz yoktu. Birkaçımız (mesela ben :)) daha yemeğini bitirmediği için bu teklif kabul edildi. Çayların gelmesi en az yarım saat sürdü. Meğer bizim için özel demlemişler. Ne zaman bıkıp hesap istesek garsonun tepkisi aynı idi:"Ama biz sizin için demledik çayı!" Sonunda çaylar geldi, içip partiye geri döndük ama hava da iyice soğumuştu parti mekanına gidene kadar soğuktan feci şekilde dondum. O saatten sonrada sabaha kadar kendime gelemedim. Bir ara sızmıştım sağolsun Decipher o sırada geçti kodun başına birkaç şeyi tamamladı. Zaten hastaydım üstüne gece yediğim iyice yorulmuştum. Ancak oyunu tamamlayacak enerjim kalmıştı. Scroll Compo'ya katılmayı da düşünüyordum ama artık enerjim kalmadığından vazgeçtim. Sabah 12'ye doğru oyun bitti ve Windows'ta derlemeleri için (çünkü benim laptop'ımda Linux yüklü idi) Tesla ve Decipher'a teslim ettim kodu. Onlar da sağolsunlar derlediler ve ben de organizatörlere oyunu teslim edip kendimi çevrede dönen geyiklere verdim.

İşte uyku tulumu ve şişme yatak teknolojilerinden bihaber, tecrubesiz bir parti ziyaretçisi incecik bir battaniye ile uyurken

Şekil 9.

The Day After That Day:

2. gün artık oyunu bitirip teslim ettiğim için oldukça rahatlamıştım ve çevrede dönen geyiklere verdim kendimi iyice. Bu sırada zaten en başından beri şüphelendiğimiz gibi katılım azlığından compoların iptal edildiği haberi geldi. Zaten beklediğimiz için pek umursamadık. Akşama doğru submit edilen productlar dev ekranda izlendi (ve dinlendi). Yanlış hatırlamıyorsam 2 şarkı, 2 grafik, 2 scroll intro ve 2 oyundan ibaretti katılanlar. Geliştiriciler alkışlandı, tebrik edildi, ürünler beğenildi ve parti sona erdi. Herkes yavaş yavaş toplandı, birkaç kez vedalaşıldı. En son olarak toplu birkaç fotoğraf çekildi.

Eve Dönüş:

Geldiğimiz kadro ile toplanıp önce Beşiktaş'a gittik, orada yemek yedikten sonra vapurla Kadıköy'e geçtik ve oradan da Haydarpaşa'ya gittik. İçilen çaylar ve yapılan geyiklerden sonra sonunda trenin kalkma vakti geldi ve yerlerimize geçip Ankara'ya dönüş yolculuğuna başladık.

Demoscene'de yeni konsol: GP2X

Atilla Filiz

2005 sonunda yeni bir cihaz, demo camiasına güçlü bir giriş yaptı. Gelen, GamePark Holdings adlı Koreli bir şirketin kişisel eğlence konsolu *GP2X* idi. Bu cihaz, PSP ya da NDS gibi *piyasa* cihazların aksine demoscene gibi tam bir *underground* oyuncağı.

Burada teknik ayrıntılara girmek istemiyorum, zira teknik bilgileri bulmak çok kolay. Burada daha çok, bu güzellik hakkındaki şahsi fikirlerimi paylaşacağım.

Her şeyden önce bu cihazı piyasaya sürenlerin(GPH) amacı, kullanıcılar arasında bir cemaat oluşturmak ve kullanıcıların bir şeyler geliştirmesine destek olmak. Bunun için *linux* tabanlı bir işletim sistemi kullandılar ve daha önemlisi, yazılım geliştirme kitleri ile bazı belgeleri kullanıcılara sundular. Sonuç, kısa bir süre içinde başka platformlarda(genellikle pc) olan oyunların GP2X'e taşınması ve bir sürü de emülatörün geliştirilmesi oldu. Ayrıca neredeyse her partide en az bir iki GP2X demosu da sunulmaya başladı.

Şekil 1. GP2X kişisel eğlence konsolu

İthalatçı şirket sağolsun, Türkiye'de de bir GP2X topluluğu oluşmuş durumda. gp2xtr forumları [http://gp2xtr.com] ve wiki sayfalarını, ithalatçı şirket olan *SMart* oluşturdu ve destekliyor.

Şekil 2. Smart firmasının Nightshift 07 Demo partisindeki standı

Ben GP2X ile, 2006'da Compex Dijital adlı fuarda tanıştım ve o gün vuruldum. Herşeyden önce, bu donanım, gelişmek için tasarlanmış. Alt tarafında bulunan EXT adlı konnektörün içinde iki adet *USB* kapısının yanı sıra *TV çıkışı* ve geliştiriciler için bir seri konsol ile bir *JTAG* portu var. Diğerlerini geçelim, USB sayesinde aklınıza gelen her şeyi bu alete bağlayabilirsiniz. Klavye ve farenin yanı sıra her türlü depolama birimi(flaş bellek, kart okuyucu, hard disk vs.) zaten destekleniyor. Kullanıcıların geliştirmesiyle ayrıca Bluetooth,wifi gibi çevre birimler de henüz tam destekle olmasa da kullanılıyor. Konsol ve JTAG arayüzleri ise geliştiricilerin işini büyük ölçüde kolaylaştırıyor, çünkü bu çıkışlar sayesinde yazılımlarda hata ayıklamak ve bazen de(benim başıma geldi) arıza gidermek mümkün.

Açık olan belgeler ve geliştirme kitleri sayesinde ise bu şey tam bir demo cihazı. Donanımı 3B değil ama 2B hızlandırmaları destekliyor. Eğer yapacaksanız, 3B efektler sizin yeteneğinize kalmış. Her biri 200'er MHz olan bir ana + bir yardımcı işlemci bir çok işin altından kalkıyor. Günümüzün masaüstü bilgisayarları kadar işlemgücü ve kolaylık sunmuyor ama eski bilgisayarlardan da çok daha büyük kolaylıklar sağladığı kesin.

Unutmadan, bu şey piyasa bir oyuncak olmadığından, GP2X için pek piyasa oyun beklemeyin. Niid For Spiid kıl, Hayri Pıtır yün, Faynıl Fentezi 22 gibi şeylerin(en azından hepsinin) GP2X için çıkması pek muhtemel değil. Zaten yapılış ve piyasaya sürülüş amacı bu değil. Bunu da belirtelim.

Sonuç olarak GP2X, kendisi bir şeyler geliştirmeyi seven, eski bilgisayarlara veya konsolara ilgi duyan, 2B oyunları seven ve demo seven kişilere hitap eden çok şeker bir konsoldur diyebiliriz.

Müzik İncelemeleri

Bilgem 'Nightlord' Çakır

Nightshift 07 Müzikleri

Müzik Yarışması

Öncelikle birşeyi yanlış yaptığımı itiraf ederek başlayayım. Bu reviewleri partiden hemen sonra sıcağı sıcağına yazmalıydım. O sırada bu parçaların üzerimdeki etkileri daha yoğun ve şaşkınlık vericiydi. Şimdi aradan zaman geçtikten sonra bu eserlerle ilgili değerlendirmelerimde haklarını veremeyeceğimden endişeliyim. Bu yüzden öncelikle değerlendireceğim parçaları yapan arkadaşlardan özür diliyorum.

Müzik yarışması genel olarak çok sayıda üzerinde özenilmiş parçaya ev sahipliği yaptı bu yıl. Ben özellikle dikkatimi çeken dört parçaya değineceğim. Bu parçaların hepsini http://nightshift.untergrund.net adresinden indirebilirsiniz.

İlk değinmek istediğim parça *Punky*'nin *Infusion* adlı parçası. Bu benim dinlediğim ilk Punky parçasıydı ve çok pozitif bir şekilde beni etkiledi ve şaşırttı. Daha girişteki filtreli synth sesler ve arkasından giren ritm çok hoş, yalın ve profesyonel. Ardından giren yumuşak armoni bölümü gerçekten beni havaya sokuyor. Üstelik ritmde yer yer serpiştiriliş senkop bölümler de parçaya dinamizim katıyor. Bunun yanında özellikle 00:55'teki ritm kesilmesi ve 1:12'deki synth arpejin kısık sesle girmesi çok hoşuma gitti. Zaten aynı arpej daha sonra 1:36 da daha yüksek volume ile geri geliyor ve bizi parçanın sonuna taşıyor. Parçanın 2 dakikadan biraz uzun olan süresi de tam tadında bırakılmış.

Şekil 1. Punky

Punky'nin özellikle sound konusunda çok iyi ve yetenekli olduğunu düşünüyorum. Bundan sonra yapacağı müzikleri heyecanla bekliyor olacağım. Bir diğer merak ettiğim konu da Punky'nin sonraki parçalarında nasıl tarzlara uğrayacağı. Genel olarak *groove elektronik* mi devam edecek, yoksa başka türlere de bir miktar yayılacak mı? Nasıl bir yelpazede ürünler çıkaracak. Özellikle hem sound ekseninde hem armoni ekseninde şapkasında ne tavşanlar saklıyor acaba :) Dediğim gibi heyecanla bekliyor olacağım.

İkinci değinmek istediğim parça *Drey*'in *Sidling* adlı eseri. Drey'in demodojo soundtrack'i hakkında daha önce zaten nasıl bir şaheser olduğuna dair yorumlarım olmuştu forumda. Bu yorumları bu yazının ardına da ekleyeceğim çünkü Plazma daha kalıcı bir platform ve daha çok kişiye ulaşıyor.

Sidling farklı bir kulvardaki bir eser. Onu *Demodojo soundtrack* ile karşılaştırmak haksızlık olur (hatta benim için Demodojo soundtrackle karşılaştırılabilecek çok çok az eser var).

Sidling tam bir Parti kompo parçası aslında. Akıcı, hızlı iyi melodiler, insanı yakalayan bir ritm ve gaz akor yürüyüşleri. Elektro synth ritm arpejler ve melodiler ile parçanın özellikle ikinci yarısında giren hardrock bateri başarıyla birbirine entegre edilmiş. Bunun yanında parçanın her yerine hakim bir teknik sound ve mixing ustalığı yine göze çarpıyor. Yine parçanın geneline hakim synth lead sesteki akışkanlık ve ustalık dikkatimi çekti. Parça genel olarak giriş gelişme ve sonuç bakımından da iyi kurgulanmış.

Sonuç olarak hakkında söylenebilecek hiçbir negatif şey yok. Sadece Demodojo soundtrack kadar direk *omurilikten bünyenize hükmetmiyor* :). Drey'in ne kadar iyi bir müzisyen olduğuna dair zaten yakın ve güvenilir kaynaklardan duyumlar almıştım. Fakat *NS07*'de eserleriyle karşılaşınca gerçekten tren çarpmışa döndüm. Bakalım bundan sonra neler yapacak...

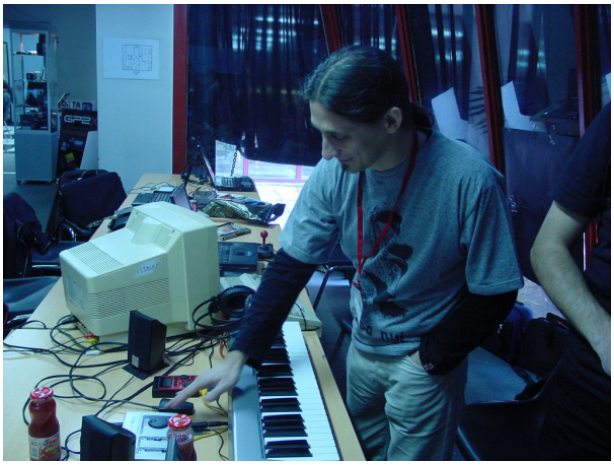
Müzik compodan üçüncü değerlendireceğim parça ise *Schimmer and Fraud*' dan *Afromope* adlı parça. Benim çok az farkla bu compodaki favorim bu parça. Özellikle 1. dakikada giren saksafon kullanım çok ustaca ve beni benden aldı. Aynı şekilde 1:46'dan itibaren geri gelen saksafon melodisi benim gibi genelde doğu batı sentesi çalışmalara çooook şüpheyle yaklaşan biri için bile çok dürüst ve hoştu. Ayrıca parçanın başında ve sonunda kullanılan jazz gitar sound'u da çok hoşuma giden deatylardan biri. Zaten yer yer kullanılan 70'ler diskosu tarzı bas yürüyüşleri ile de hoş bir entegrasyon sağlamış. Hoş ve nispeten karmaşık bir elektronik ritm - perküsyon bileşimi var. Armonik yönden çok minimalist bir parça. Ama bas, yerinde kullanılan efektler, yalın bir doğulu saksafon yorumlanışı bu kadar güzel bir örgü kurabilirdi diye düşünüyorum. Dediğim gibi benim kişisel olarak pek tercih etmediğim ve hoşlanmadığım pekçok öğe barındırmasına rağmen bu parçanın bütününe bayıldım.

Şekil 2. Wisdom

Muzik compoyu yine tepede tamamlayan ise Wisdom'dan gelen Winter Syndrome. Parçayı dinler dinlemez o gitar sesine tav olmamak elde degil. Sanırım parçanın yarışmada birinci olmasının en büyük sebebi bu gitar sesi ve çaldığı çok özgün melodi. İhtimalen Wisdom'ın kendisi parçanın henüz tamamlanmadığını bütün enstrümanların cilalanmadığını düşünüyordur (partilerde yaptığı parçalar hakkında hep böyle düşünür)

Şekil 3. Flexi

Bu sene yarışmada gözlerimizin (ve kulaklarımızın aradığı) iki isim vardı. Biri Flexi/Resident digeri ise Impetigo/Crescent. Oldukça özgün tarzları olduğunu düşündüğüm bu iki müzisyenin yeni çalışmalarını da heyecanla bekliyorum



Şekil 4. Impetigo

Demodojo Soundtrack

NS07'de müzik componun dışında beni çok etkileyen bir parça vardı. Bu *Demodojo'nun* ilk demosunun müziğiydi.

Parçanın başlangıcını ilk duyduğumda japon motifinin melodiye yedirilişindeki ölçülülük dikkatimi çekti. Çünkü bazı etnik motifler var ki batı müziğinde kullanılışları bazen çok sıradan ve samimiyetsiz olabiliyor. Belki bin tane yapıt var böyle aynı orta doğu gamını veya uzak doğu gamını kullanan. Batılılar sanıyor ki, o gamlarda melodi yazdılar mı tamam kültürel füzyon oldu. Bu da çoğu zaman o kültürlerde yer alan bilgisizliğin ve ilgisizliğin bir müzisyen üstündeki yansıması olmaktan öteye gidemiyor.

Oysa *Drey*'in melodisi eserin türü olan batı klasik müziğine, abartmadan ve aşırıya kaçmadan tam olması gerektiği ölçüde bir uzak doğu motifi ve çok minimal bir armoni katmakla yetiniyor. Ardından giren keman ritmi bütün parçada bana biraz rahatsızlık veren tek yer (5. ve 8. ölçüler arası). Ama parçada hiçbir gereksiz tekrar olmadığı için 9. ölçüden itibaren giren daha bas yaylılarla birleştiğinde artık karşınızda sıradan bir orkestrasyon olmadığını anlıyorsunuz.

Ardından ustaca gelişen bir bas yürüyüşü, saldırgan bir davul ritmiyle kalp atışlarınızı hızlandırmaya başlıyor. Belki de *Demodojo'nun asıl engellenemez yükselişi* buradaki 8 ölçülük gelişme. Arka planda bunlar olurken önde melodi uyumlu bir şekilde saldırganlaşıyor. Burada dikkatimi çeken çok önemli iki nokta var. Birincisi çanların tuttuğu ritm. Yine ne eksik ne fazla. Tam tadında ve çok zenginlik katıcı bir dokunuş. İkincisi ise tam arp geçişinden önceki brass section'ın yaptığı kısa crescendo. O kadar ustaca yapılmışki oradaki enstrümanların seçimi ve volume kontrolü, gerçekten Londra Filarmoni mi bu diye şüpheye düşüyor insan. Bilgisayarla yapılan müziklerde bana göre en zor en ustalık isteyen konulardan biridir brass section. Biraz müzik bilgisi olan herkes yaylıları kolayca kullanabilir. Fakat bir kompozitörün brass kullanımı bence en büyük ustalık göstergelerinden biridir. Drey bu konudaki ustalığı ile zaten parçanın ileri bölümlerinde bizi fazlasıyla büyüleyecek ama buradaki bir saniyelik ilk kullanım bile pek çok kompozitörün beceremeyeceği birşey.

Ve Harp geçişi... Her zaman orkestrasyonlarda hoşuma giden bir detaydır. Burada Drey brass section crescendosunun ardından yaylıların armonisinde sadece en tiz notayı bırarkırken önünde harpı kullanıyor ve dinleyiciyi öyle bir noktada bırakıyor ki...

Şekil 5. Drey

Do La... Sol re mi... Bir anda nefesimi kesen piyano melodisi. Öyle yalın, öyle samimi, öyle dokunakli... Sol el piyano da Fa Major - Sol Major - La Minör arpeji ile o yalın melodiye eşlik ediyor. Daha burada piyanonun büyüsünden kurtulamamışken La minörle beraber giren violaların arpeji bütün benliğimi teslim alıyor. Hiç birşey kendini tekrar edemiyor. Bir de bakıyoruz ki yeni bir gelişme sekansına girmişiz bile. Hem de olaylar o kadar hızlı ve dinamik gelişiyor ki. O piyano melodisine gireli henüz 4 ölçü olmuşken violaların arpeji başlıyor. 6. ölçüde bu arpej çift sesli hale geliyor. 8. ölçüde o muhteşem brass melodisi, tüylerimi diken diken eden askeri trampet ritmiyle beraber katılıyor bu ustaca hazırlanmış dokuya. Drey'in sanatçılığına, kurduğu müzikal cümlelerin tetiklediği duygu yoğunluğuna mı odaklanayım, bütün bunları yaparken gösterdiği teknik ustalığa mı (hiç bir enstrümanın doğal sınırlarının dışında kullanılmaması, volume mixingindeki inanılmaz temizlik, nota vuruşlarındaki şiddet dinamizmi) odaklanayım bilemiyorum. Her nota her bir enstrümandan çıkabileceği en temiz ve yalın haliyle çıkıp karmaşık ama kusursuz bir uyumdaki bu müzikal dokuda kendine ait yeri alıyor.

Ardından gelen 8 ölçü ise daha az gözönünde olan bir harp geçişiyle başlıyor. Bu sefer Drey orkestrasındaki enstrümanların rollerini değiştiriyor. Piyano arka plan arpeji ile geri dönerken, tiz yaylılar 4 ölçü boyunca bizi brass section ile başbaşa bırakıyor. Burada benim çok hoşuma giden bir olayda bu 8 ölçü boyunca çok hakim bir melodi olmaksızın sadece armonik bir ilerlemenin parçayı domine etmesi. Benim gibi kontrapunto severler için çok lezzetli olan bu bölüm son 4 ölçüde yaylıların brass section ile karşıt yükselip alçalan çizgisi ile zenginleşip bizi herşeyin başladığı yere doğru götürüyor.

Sonraki 8 ölçüde piyanodan önceki ilk yükselişte tanıştığımız melodi ile tekrar buluşuyoruz. Bu sefer ilk seferkinden farklı bir armonik yürüyüş içinde dolayısıyla daha farklı, hatta daha olgun ve belki biraz da vedalaşmaya hazırlanan bir örgü içinde. Zillerle yapılan bir crescendo bütün orkestrayı sessizliğe yöneltirken bizi tekrar ana melodi çanlar ve bas yaylılarla bırakıyor.

Parça bittiğinde nasıl bir yoğunlukla, ustalıkla ve ölçülülükle karşılaştığımı nasıl bir şahesere maruz kaldığımı bütün benliğimle hissettim. İşte bir sanatçının teknik temellerindeki sağlamlığı, sanatsal ifadesindeki özgünlüğü ve dürüstlüğü birleştireceğine dair inanılmaz bir örnek. Zaman zaman bu mükemmel birleşimi yakalamanın hiçbir zaman mümkün olamayacağına inanıyorken, her 4-5 yılda bir karşıma çıkan bir eser, beni bu derecede etkileyip, bu mükemmelliği hep aramamız gerektiğini ve belki birgün yakalayabileceğimizi tekrar öğretiyor. Drey'e, herşeyden önce bana teknik ve sanatsal mükemmelliğin mümkün olduğunu tekrar öğrettiği için teşekkür ediyorum. Scene'e girdiğin o kutlu güne ve buna dolaylı olarak aracı olan Plazma dergisine ayrıca teşekkür ediyorum.

Scene bu ülkede ne zaman ne iniş çıkışları yaşayacak bunu kestiremiyorum. Ama verilen organizasyonel ve yayınsal emeklerin karşılığında hiçbirşey olmasa bile sadece Drey'i kazanmamız ve onun bu şaheseri yaratıp bizlerle paylaşmasına meydan sağlamamız bile şahsen bana yeter...

7d7 Müzikleri

Bu parçaların hepsini http://7dx-party.org adresinden indirebilirsiniz. Burada değindiğim parçaların yanında Chaotique/Demodojo City of Jaded Eyes isimli bir parça yayınlarken, Crescent'ın iki usta müzisyeni Wisdom ve Impetigo da 7D7 Jam isimli ortak bir çalışma yayınladılar.

Machine's Dream - Drey/Demodojo

Bu benim için hayli gaz ve ilginç bir parçaydı. Benim gibi 80 sonu 90 başı gitar riffleri ekolünden gelen biri için özellikle parçadaki ritm gitar kullanımı çok güzel bir unsurdu. Aslında parçayı daha dikkatli incelediğimde zaten benim beğenmemi sağlayan bazı başka unsurları da görebiliyorum.

Bir kere Drey'in genelde seçtiği akor dizilimleri tam benim zevkime göre. Bol bol gaz La minör Fa majör geçişleri olsun, interlude'deki Fa majör, Sol majör, La minör, Mi minör dizilimi olsun, bunlar teenage rock'çı yıllarımızda kanımıza işlemiş bir kere... Şimdi duyduğumuz yerde kafa sallayıp *power stance* durumuna geçiyoruz (power stance ile ilgili olarak bakınız school of rock filmi)

Zaten parçanın ilk iki dakikasındaki ritm gitar temellerinin üzerine interlude'den önceki 4 ölçülük solo da iyice beni benden alıyor. Ben olsam orada daha en az 12 ölçü daha gitar öttürmeye devam ederdim bu gazla... Ama işte ölçülü insan Drey tadı damağımızda bırakmasını birkez daha beceriyor.

Ama Machine's dream sadece gitarlardan ibaret bir parça da değil. Baştaki piyano introsu, 3. dakika civarlarında giren sentetik seslerle yapılan ınterlude bölümü, parça boyunca bas gitarın gösterdiği değişimler hep yine çok ustaca. Özellikle bas gi-

tardaki duru ama sağlam melodiler parçaya çok önemli bir referans noktası sağlıyor. Zaten akor diziliminin arkasında sabit akorda kalan bas kullanımı hep hoşuma gitmiştir, bu parça da en iyi örneklerinden biri.

Son olarak yine parçanın genelindeki giriş, gelişme, sonuç akışı ve seslerin ve *mixing*in teknik kalitesine hayranlığımı dile getirerek bu güzide parça ile ilgili düşüncelerimi tamamlayayım. Yüce yetenek Drey'in hastasıyız ailecek ve her parçasını heyecan ile beklemeye devam ediyoruz.

Şekil 6. Hydrogen

Drey

Şarkıyı ve ismi geçen müzisyenleri tanımayanların bir isim kargaşası yaşamaması için ilk başta bazı açıklamalar yapayım. Drey aslında bir FRP hikayesinde bir karakterin adı idi. Ardından, Hydrogen adlı C64 müzisyeni, bu karaktere ithafen Drey isimli bir SID müziği yaptı. Ardından scene'e yeni ilgi duyan bir diger müzisyen kendisine scene takma adı olarak Drey'i seçti. Sonra da Drey takma adlı bu müzisyen Hydrogen'in Drey adlı SID müziğine pc'de remix yaptı. Hala kafanız karışmadıysa şimdi karışacak. Drey karakterini ilk yaratıp, FRP hikayesine koyan da zaten Drey (müzisyen olan) idi. :)

Bu nasıl orjinal bir orkestrasyondur. Etnik, senfonik, metal nasil bu kadar guzel dengelenir. Bir kez daha akıllara durgunluk verecek bir iş ile karşı karşıya kalmanın tarifsiz dumur ve mutluluğunu tadıyorum.

Önce Drey vardı. Kadim zamanlardan beri yaşayan bir irfan. Sonra Hydrogen ona bir SID besteledi. Şimdi Drey o SID'i başka bir medyada canlandırdı. Biz normal ölümlüler de bu yetenek pınarı soydan daha başka neler çıkacak diye bekler olduk.

Öncelikle şunu itiraf etmeliyim. Ben Drey'in SID versiyonunu resmen anlamamışım. Yani kendimi hayret ve dehşetle kınıyorum. Orkestrasyonu dinledikten sonra orjinali yeniden dinlediğimde zaten orada olan bazı temel armonik öğeleri resmen tamamen kaçırmış olduğumu farkettim. Bunun için daha önce Drey SID ile ilgili olarak Hydrogen'e saygı ve tapınmalarımı yeterince iletmemiş olmaktan dolayı utanç içindeyim.

Parçanın orjinal kompozisyonunda yer alan 50. saniyedeki akor değişimi aradaki interlude ve ınterlude sonrasındaki solo melodi (orkestra versiyonunda elektro gitarın seslendirdiği) harika öğeler ve Hydrogen'e şapka çıkartıyorum.

Orkestrasyonda ise Drey bu esere hakkettiği kalitede bir aranjman ve teknik mükemmellik ile yeni bir boyut getiriyor. Yine günlerce parçayı tekrar tekrar dinleyerek bu iki ustanın uyguladıkları trikleri tespit edip öğrenmeye çalışıyorum.

Drey orkestrasyonda parçayı biraz daha yavaşlatarak başlamış işe. Bu hem bas ritmi taşıyan çello/kontrbas grubunun daha gerçekçi ve doğal duyulmasını sağlıyor, hem de daha sonra giren slow heavy metal bateri ritminin daha patlayıcı bir etki yaratmasını sağlıyor bana göre.

Beni benden alan bir diğer trik 4.ölçüde giren tef + üçgen zil karışımı etnik perküsyon. Ben daha buna diyecek söz bulamazken 8.ölçüde giren darbukaların etnik ritmi güçlendirmesi ve aynı anda önde yine brass section'ın melodiye girişi, 12. ölçüde ritm gitar ve baterinin girişi, 16. ölçüde lead gitarın girişi 18'de modülasyon ve ardından süper bir geçiş ile Hydrogen'in harika interlude bölümünün oboe, yaylılar ve hafiflemiş etnik perküsyon ile girişi. Sanki arka arkaya birbirinden güzel resimlere bakıp daha birindeki detayları inceleyemeden diğerinin gelmesi gibi bir his yaratıyor bende. Bu parçanın özellikle ilk 26 ölçüsünün kompozisyon derslerinde build-up örneği olarak okutulması lazım, yok böyle bir akış. Zaten interlude'den sonraki patlayıcı final solosu ve son bölümde ritm lead ve yaylı basların birleşimi böyle muhteşem ve görkemli bir orkestrasyon örneğini de harika tamamlıyor.

Bunun ardından bize kalan bu iki üstada bol bol ilham dilemek ve "yok mu şöyle başka kuzen, yeğen, dayıoğlu, amca kızı" diye sormak oluyor

Bu tip bir eserle karşılaştığım zaman birbiri ile çelişen iki dürtü hissediyorum. Birincisi, "Ne kadar hayran kaldığımı ifade etmeliyim. Yok bu yazdıklarım yetmedi bir paragraf daha yazıp falanca unsura olan hayranlığımı da dile getireyim". İkincisi ise "Böyle bıdı bıdı yazıp durmayayım parçanın ne kadar aşmış olduğu zaten gün gibi açık, ben kimim de bu kadar review yazmaya hak görüyorum kendimde".

İşte bu iki dürtünün ortasında bir yerleri tutturmaya çalısıyorum ama tutturamama ihtimalimi göz önüne alarak açık açık yazayım dedim: Burada yazabileceğim herşeyin ötesinde bir miktarda bu parçaya hayranlık duymaktayım. Bu parçaya ve bunun gibi gelecekteki parçalara umarım benden çok daha ehil insanlar da geri besleme verirler.

Röportaj: Quiss

Emir 'Skate' Akaydın

Şekil 1.

Bir haftalık bir çabanın ardından Quiss/Reflex ile kontağa geçmeyi başardım. Quiss kimdir, necidir, scene'de neler yapmıştır bunların hepsi röportaj içersinde açık ve detaylı bir şekilde yer alıyor. Ancak bir ön bilgi edinmek isterseniz aşağıdaki linkten Quiss'in CSDB'deki profilini ve yayınladığı ürünleri inceleyebilirsiniz. Elbette ki CSDB'deki tarihçede eksikler var ama Quiss'in en önemli demo ve oyunları listede yer alıyor.

Quiss'in CSDB'deki profil linki:

http://noname.c64.org/csdb/scener/?id=844

Online Röportajın Soru ve Cevaplarına geçelim...

Skate: Merhaba Matthias, öncelikle bu röportajı yapmayı kabul ettiğin için sana çok teşekkür ederim.

Quiss: Birşey değil.

Skate: Sen scene'i bırakalı yaklaşık 10 sene oldu ve ardında birçok sıkı demo, oyun, koleksiyon ve tool bıraktın. Bu röportajı yapmamdaki öncelikli amacım seni yeni Türk scenerlara tanıtmak, özellikle de Commodore 64 platformunda programlama ile ilgilenenlere. Hadi klasik soruyla başlayalım. Bize biraz kendini tanıtabilir misin? İsmin, yaşın, doğduğun/yaşadığın ülke vb.

Quiss: Gerçek adım Matthias Kramm. Almanya'da doğdum ve halen Munich yakınlarındaki "Garching" isimli küçük bir Alman kasabasında yaşıyorum. 29 yaşındayım.

Skate: Bildiğim kadarıyla scene kariyerine 1993 yılında başladın. Ancak tahmin ediyorum daha öncelerinde de bilgisayar kullanıyorsundur. Tam olarak ne zaman bilgisayar kullanmaya başladın? Commodore 64 ilk bilgisayarın mıydı?

Quiss: C64 benim ilk bilgisayarımdı ama programcılığı daha önceleri yaz okulunda öğrenmiştim. "LOGO" isminde (yanılmıyorsam) eski Apple bilgisayarlarıyla çalışıyorduk. 1986 yılı civarlarıydı. C64'ümü bir iki yıl sonrasında almıştım. Tam zamanından emin değilim ancak eski disk koleksiyonlarımda (hala duruyorlar, PC'imde de yedekleri mevcut) 1988 tarihlerine rastlıyorum.

Genç Matthias (Quiss) Commodere Basic'iyle oynaşırken. :)

Şekil 2.

Skate: Programlamayı yaz okulunda mı öğrendin? Çok ilginç. Özellikle de o yıllar için. Peki programlamaya Basic diliyle mi başladın?

Quiss: Evet, elbette ki. Herkes Basic ile başladı. Hala elimde iki disk dolusu her türden cool basic programları duruyor (kayan yazılar (8'er pixel, henüz D016 kullanmadan), parlayan borderlar (POKE 53281), vs. :))

Skate: "Herkes Basic ile başladı", eğer 1990 yılından önce doğdularsa. :)

Quiss: Elbette ki. Programlamaya Javascript ile başlayan, C64'ün 64-Bit C Compiler'ı olduğunu sanan yeni jenerasyondan bahsetmiyorum. :)

Skate: C64 scene'ine nasıl bulaştın?

Quiss: Aslında bulaşmadım, yani başlangıçta. Programlamaya oyun yazarak ve onları 64'er magazine'e satarak başladım. "Game On" ve "Magic Disk" C64 magazinleri sayesinde scene hakkında ucundan kıyısından fikir sahibi oldum ve scene ilgimi çekti. Yanlış hatılamıyorsam Mamba scene magazini bir demo yayınladı o sıralar. Sonrasında ilk olarak Slash Designs grubuna üyelik başvurusu yaptım, ancak kısa süre sonra onlardan ayrıldım ve bir süre sonra da Reflex'e katıldım (Reflex ile "Nova" isimli oyunumu gördükten sonra benimle irtibata geçen PVCF aracılığıyla kontağa geçtim)

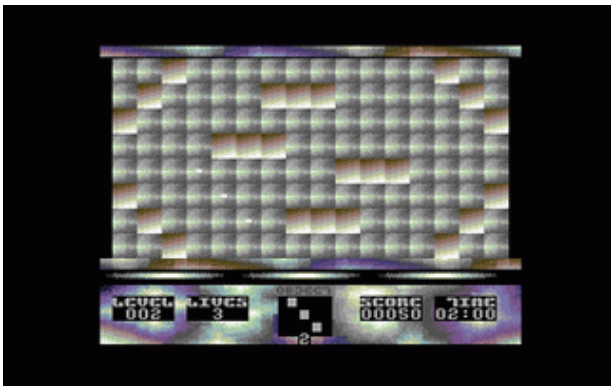
Skate: Demek scene kariyerine oyun programcılığıyla başladın. İlk olarak Muehle isimli bir oyun yayınladığını biliyorum. O oyundan önce başka oyun projelerin var mıydı?

Quiss: Aslını sorarsan Muehle zincirin halkalarında epey ilerde kalır. İlk oyunum (aynı zamanda ilk yayınlanan oyunum) "Boom" ismindeydi (tek kişilik shoot'em'up), ve 64'er tarafından yayınlanmıştı. Daha sonra "Ace" (Tron benzeri bir oyun) ve "Cyclones" (bir strateji oyunu) oyunlarını hazırladım. Bu ikisi hiçbir zaman resmi olarak yayınlanmadılar. Ondan sonra (ve Art Project Studios isminde oyun geliştirmeye odaklı bir gruba katıldıktan

sonra) Muehle ve Nova II'yi yayınladık.

Skate: Anlıyorum. Ben şu anda biraz eksik gözüken CSDB'deki kişisel bilgilerini temel almıştım. Aslında bu soruyu sormamdaki temel amaç da buydu. Yanıtımı aldım. :) Peki Nova 2, Plopp, Geometric 2'den ne haber? Onlarla ilgili anıların var mı? Örneğin Nova 2'nin giriş ekranında "starnoter" tool'undaki tam ekran starfield efektini kullanmışsın. Biraz kamera arkası dinlemek istiyorum. :)

Quiss: Nova 2'yi yapmak inanılmaz derecede eğlenceliydi. Özellikle birçok insan o projede yer aldığı için. İki müzisyenimiz (PVCF ve Odysseus), bir grafikerimiz ("Seytan", o zamanlar "Dj Bobo" idi), ve hatta tek işi bu olan bir level tasarımcımız (Sonic) vardı.



Şekil 3.

Geometric de oldukça güzel bir projeydi, çünkü zamanının çok iyi grafikerlerinden olan Seytan ile yakın çalışma imkanı elde etmiştim ve kendisi oyunu olabildiğince iyi yapmak için oldukça gayretliydi. Hatta hatta oyun ekranlarındaki FLI grafikleri daha verimli çizebilmesi için oyuna özel bir FLI grafik editörü bile hazırlamıştık.

Bu arada Plopp tam bir oyun sayılmaz aslında. O yalnızca "first crack" yapabilmek için "orjinal" oyun arayan bir arkadaşıma iyilik olarak hızlı bir hack ile hazırladığım bir oyundu. :)

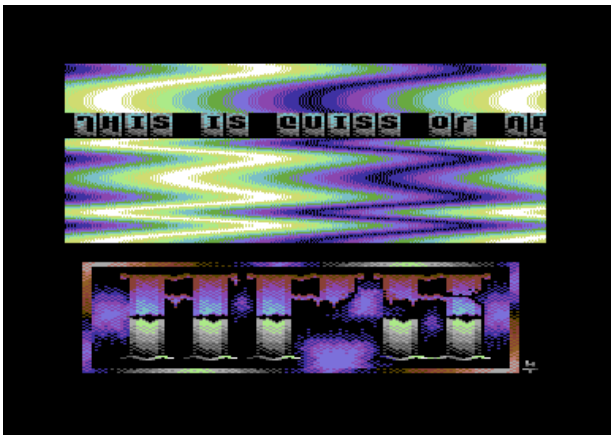
Skate: "Seytan"ın Türkçe'de şeytan anlamına geldiğini biliyor musun? (Skate: Türkçe'ye çevirince komik oldu bu soru. :))

Quiss: Evet, zaten Marc (Teufel) bu yüzden o ismi seçti. "Şeytan" Almanca'da "Teufel"in diğer bir anlamıdır ki bu da O'nun soyadı oluyor.

Skate: İlginç... Peki PVCF hakkında ne diyorsun? Kendisi çok iyi bir müzisyendir ve O'nunla çalışma fırsatı elde ettin. Onunla çalışmak nasıldı? Aranız iyi miydi?

Quiss: O çok iyi bir adamdır. O zamanlar oldukça iyi arkadaştık. Saatlerce telefonda konuşurduk. (Geçmişte şehirlerarası görüşmeler oldukça pahalıydı. Konuşmak için en iyi zaman aralığı indirimli saatler olan 02:00-04:00 arasıydı. :)) PVCF, Smiling Guru ve ben birçok ürünümüzde birbirimize çok yakın çalıştık (özellikle Radio Napalm, Access Denied ve Mathematica'da). Sömestr tatillerinde, demo ve oyunlar üzerinde çalışmak için Munich ya da Dresden'de buluşurduk.

Skate: Pekala, artık biraz senin demology'ni eşeleme zamanı geldi sanırım. :) Tek tek üzerlerinden geçelim çünkü hepsi de zamanının önemli prodüksiyonları. 13 yıl aradan sonra "Made in Heaven"ı hatırlıyor musun? :) Sideborder scroll, AFLI plazma, FLD, dalgalanan logo vs. Sanırım bu senin ilk ciddi demo programlama girişimindi.



Şekil 4.

Quiss: "Made in Heaven", Marc'ı Nuernberg yakındarındaki küçük bir kasaba olan Pleinfeld'de ziyaret ettiğim bir zaman, Marc ve benim tarafımdan yapıldı. Aslında tam bir demo olarak planlamamıştık, son bölümü sıkıntıdan o şekilde hızlıca hazırlayıverdim ve sonrasında yalnızca aptal scroll textleri yazmakla uğraştık. Diğer bölümleri tam olarak ne zaman hazırladığımı hatırlamıyorum (meeting'den önce, iirc'de vs.), ama son bölüme kadar olan kısmı da Pleinfeld'de linklendi ve küçük bir demo olarak release ettik. Demonun ismini ise Pleinfeld'de karşılaştığımız Marc'ın bir arkadaşı tavsiye etti.

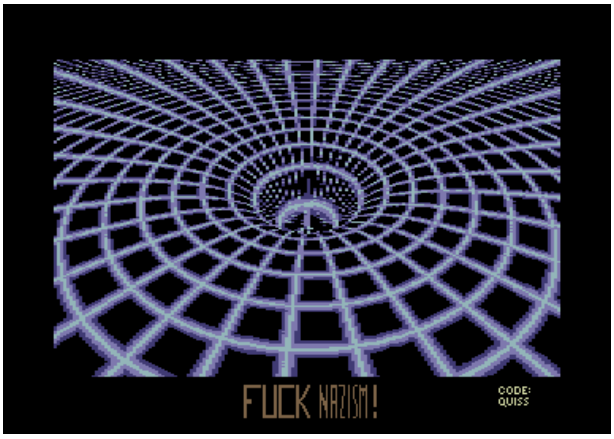
Şekil 5.

Skate: Hadi biraz da "Act Now!" anti nazi demosundan bahse-

delim. O demo büyük bir cooperationdı. Sen tam olarak nasıl bulaştın bu projeye? Ana fikir nerden çıktı?

Şekil 6.

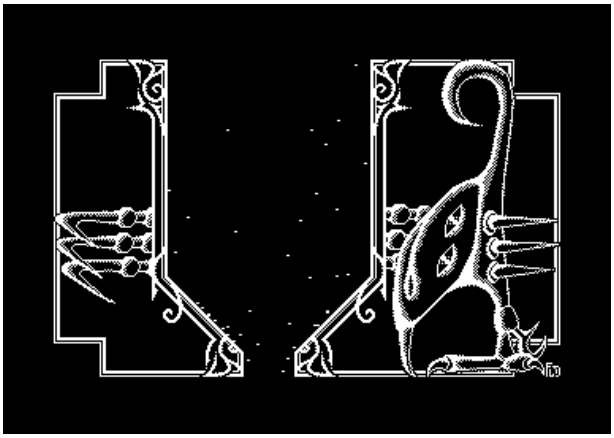
Quiss: "Act Now" projesi hatırladığım kadarıyla Oxyron grubundan Biz Kid'den çıkmıştı. Kendisi insanlara Anti-Nazi cooperation bir demoda yer almaları için bir not göndermişti. Ben de yer almak istediğimi kendisine söyledim, o da kabul etti ve gönderdiğim bölümleri demoya linkledi. Demoda yer alma motivasyonunu gamalı haç gördüğüm başka bir demodan almıştım. Aslında iyi bir demoydu ancak beni fena halde şok etmişti sonradan. Bu arada demo Byterapers'ın bir demosuydu. Biz Kid'e "bu naziler ne zaman kod yazmayı öğrendi?!" diye yazdığımı hatırlıyorum. Ancak sonrasında ortaya çıktı ki aslında demonun programcısı (Mr.Sex) bilinçsiz bir şekilde, yalnızca 3d obje olarak çevrilmesinin hoş duracağını düşündüğü için gamalı haç kullanmış, anlamını bile bilmiyormuş... Herneyse, aslında o zamanlar gerçek Nazi propagandası yapan gruplar da vardı ("KZ commander" benzeri isimli bir oyun gördüğümü hatırlıyorum), yani "Act Now" doğru zamanda yayınlanmış, iyi bir üründü.



Ed: Quiss'in demodaki partlarından biri. Bu partın müziği bu projeye katılan bir Türk scenera, Wisdom/Crescent'a ait

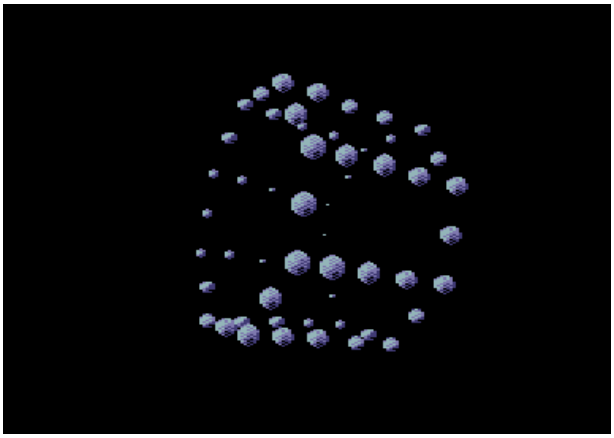
Şekil 7.

Skate: Artık scene'i biraz "sarsma" zamanı geldi! Access Denied, gerçek bir klasik. Hepimiz bu demo aracılığı ile gerçek Quiss'i tanıdık. Vektör bob, rotozoomer, plazma, güzel animasyonlu bir eor filler, zıplayan dotlar, büyük filled vektör, textured küp, landscape ve diğerleri. Hepsi de süperdi. Özellikle zıplayan dotlar bölümü çok orjinaldi, çünkü o yıla kadar programcılar demolarda çok fazla fiziksel hesaplama içeren efektler kullanmıyorlardı. O efekti kasten orjinal bir efekt ortaya çıkarma amacıyla mı yaptın yoksa bir tilt oyunu falan yazmaya başlayıp daha sonra rutinleri demoda mu kullandın?



Şekil 8.

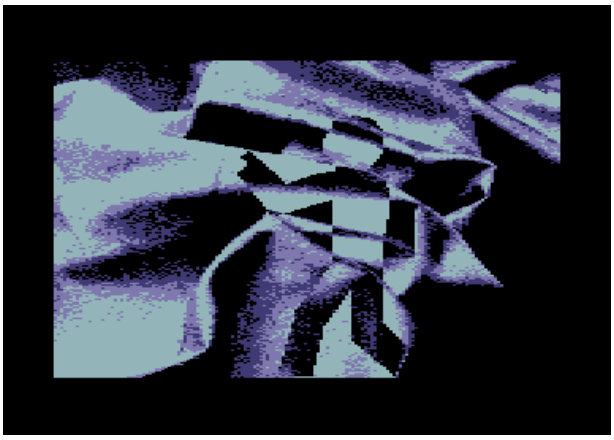
Quiss: Zıplayan dotlar bölümü Rainer (Smiling Guru) tarafından kodlanmıştı. Çok sıkı bir efektti, katılıyorum. Rainer gerçekten çok iyi bir programcıdır ve hep de yaratıcı fikirlere sahipti. Access Denied'daki diğer efektlerle gelince, kamera arkasında olup bitenlerle alakalı en çok gurur duyduğum şey, bir efektin ekranda gösterilirken diğerinin arkaplanda yüklemesi olayıdır. Buna epey bir efor harcanmıştı. Access Denied ile ilgili gurur duyduğum bir diğer şey ise demoyu bir spor müsabakasındaymışız gibi geliştirmiş olmamızdır... Demo diskini zamanında parti organizatörlerine yetiştirebilmek için "The Party"de koridor boyunca resepsiyon masasına kadar deli gibi koşmak zorunda kalmıştık. Son bölümü (PVT'nin IFLI resmi) bitiş süresinden 30 saniye önce linkleyebilmiştik. (Dev ekranda demoyu izleyene kadar doğru dürüst linklediğimizden bile emin değildik. Ancak ne mutlu ki düzgün linkleyebilmişiz. Aksi taktirde PVT büyük olasılıkla bizi öldürürdü. :))



Şekil 9.

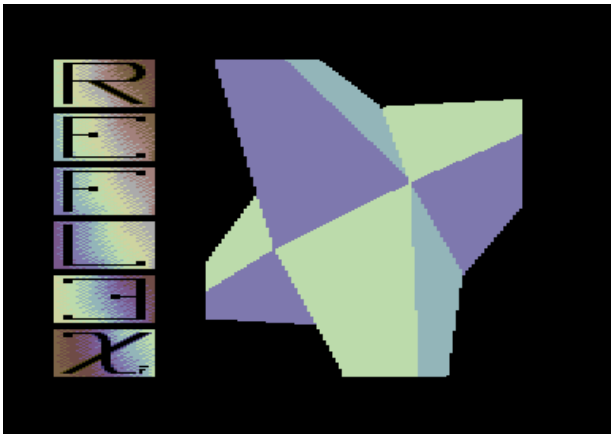
Skate: Vay be, sağlammış. :) Sanırım PVT senin aktif olduğun dönemlerde aktifti ve sen scene'i bıraktığında o da bıraktı, doğru mu? Yani hemen hemen aynı zamanlarda.

Quiss: PVT'nin ne zaman bıraktığından emin değilim. Sanırım Mathematica'dan bir süre sonra O'nunla kontağı kaybettim. Mathematica'nın yapımı sırasında bile scene ile alakalı olmayan birçok işle meşguldü zaten. Dolayısıyla sonrasında artık scene'e zaman ayıramamış olabilir.



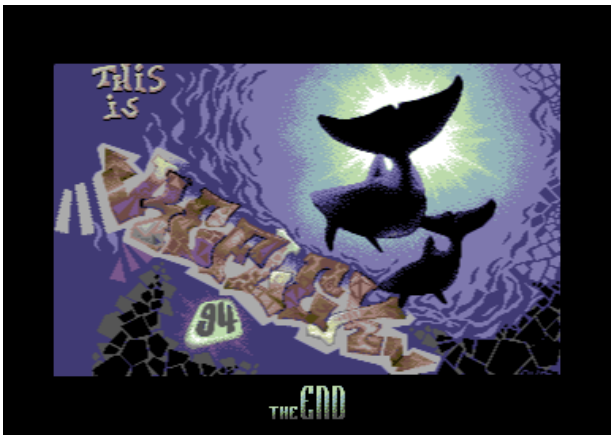
Şekil 10.

Skate: Nedendir bilmem ama yıllardan beri hep zıplayan dotlar bölümünü senin yaptığını düşünmüştüm. :) O senin değilse bile 3d bölümlerin çoğunluğu sana aitti değil mi?



Şekil 11.

Quiss: Evet, yine Guru tarafından yapılmış olan dönen texture küp hariç hepsi bana aitti. Texture verileri bellekte o kadar çok yer kaplıyordu ki, Warp 8'in çizdiği küçük grafiğin yüklenebilmesi için verileri yeterince sıkıştırabilmek hemen hemen bir haftamı aldı. Diğer bölümler için de aynısı geçerli. Access Denied için harcanmış eforda büyük pay linklemeye gitti. :)



Şekil 12.

Skate: İkinci "büyük" demon Radio Napalm'dı. Açılışta karşımıza zıplayan dotlar, "A QUISS PRODUCTION" yazısı, yine büyük bir filled vektör ve radio napalm grafiği çıkıyor. Oldukça etkileyici bir intro. Sanırım Access Denied ile bağlamak amacıyla benzer efektler kullandınız, haksız mıyım?

Şekil 13.

Quiss: Radio Napalm, Aralık 94'de yayınlanan Access Denied ile Aralık 95'de yayınlanan Mathematica'nın arasındaki boşluğu dolduruyordu... İki demonun arasında birşeyler yayınlamamız gerektiğini hissettik. Birçok bölüm ya Access Denied'ın bölümlerinin benzerleriydi (örneğin tüm vektör efektleri) ya da Mathematica'nın bölümlerinin oldukça basite indirgenmiş versiyonlarıydı (örneğin tüm tam ekran 4x4 efektleri. Bu efektler aslında Mathematica'nın "Tünel" ve "Torus'un İçinde" efektleri için kodlanmış olan 4x4 motorunu kullanıyorlardı) Radio Napalm'a özel olarak kodlanan yalnızca iki bölüm vardı. Onlar da chessboard scroller ve bitiş bölümüydü (fade eden krediler). Radio Napalm oldukça küçük bir üründü. Kimseye söyleme ama aslında sırf demo "daha büyük" gözüksün diye demo diskine random sayılardan oluşan dosyalar yaratmıştık. :) (Aslında tam olarak da random sayılmazlardı. Bir chaos fonksiyonu ile hesaplanmışlardı. x=0.21;y=0.9*4; loop: x = x*y*(1-x); write(x*256);loop end)

Şekil 14.

Skate: Hahahaha. Bu gerçekten de komikti. :) Herneyse, introdan sonra, yine orjinal bir efekt görüyoruz. Başlangıçta standart bir color cycle efektine benziyor, ancak birkaç saniye sonra görüyoruz ki harita 3 boyutlu dairesel bir şekilde dalgalanıyor. Bize efektle ilgili birkaç ipucu verebilir misin?

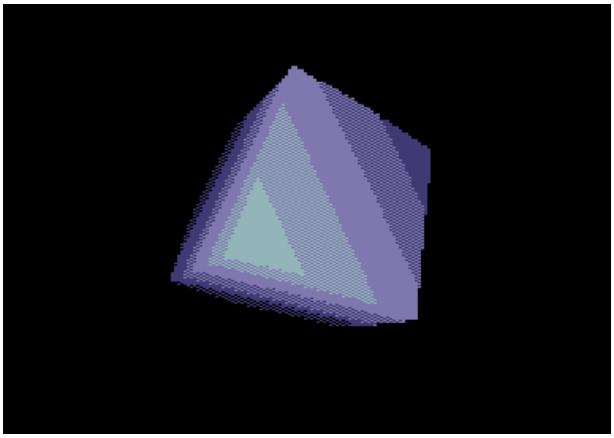
Şekil 15.

Quiss: Dalgalanan elipsler zamanına göre hoş bir efektti, ama sanırım biraz fazla karmaşık bir yöntemle kodlanmıştı. Sanırım Smash Designs'dan AEG sonraları çok daha optimize ve hızlı bir uyarlamasını yapmıştı bu efektin.

Geçmişte C64 ile ilgili en güzel şeylerden biri de buydu. İnsanlar birşeyler bulurdu, diğerleri bunu geliştirirdi, ta ki C64 donanımındaki en son damlayı sıkıp suyunu çıkarana kadar.

Skate: Çok güzel. Peki ön hesaplamalar ve harita üretme gibi konularda çapraz geliştirme kullanıyor muydunuz?

Quiss: Rainer ve ben kendi C64 cross-assembler'ımızla "yalnızca" çapraz geliştirme kullanıyorduk. Bu geliştirme sürecini büyük ölçüde kolaylaştırıyordu. Aynı zamanda, hemen hemen tüm ön hesaplamalar PC'de yapılıyordu. Genellikle, yalnızca müzik ve grafikler doğrudan C64 üzerinde hazırlanıyordu. Mathematica'nın bir kısım grafiklerini de (PVT'nin Mandelbrot fractal'ın önündeki grafiti logosu gibi) doğrudan PC üzerinde hazırladık. Swap yaptığım dönemlerde, disk notlarımı da C64 yerine çapraz geliştirme kiti kullanarak PC üzerinde yazıyordum.



Şekil 16.

Skate: (Ben o dönemde Turbo Assembler bile kullanmıyordum henüz. :)) Bu efektlerin ardından, bir diğer oldukça büyük ve hızlı bir vektör görüyoruz, bir uzay gemisi.. Filled vektörlerle ilgili sırrın nedir? Poligonları nasıl çiziyordun ve erkanı nasıl temizliyordun? Eğer senin için problem değilse 10 yıldan fazla bir sürenin ardından bunları bizimle paylaşır mısın? :) Elbette ki geçen yıllar içersinde daha hızlı vektörler elit programcılar tarafından yapıldı. Ama senin vektörlerin hala ilk 10'a girer, orası kesin.

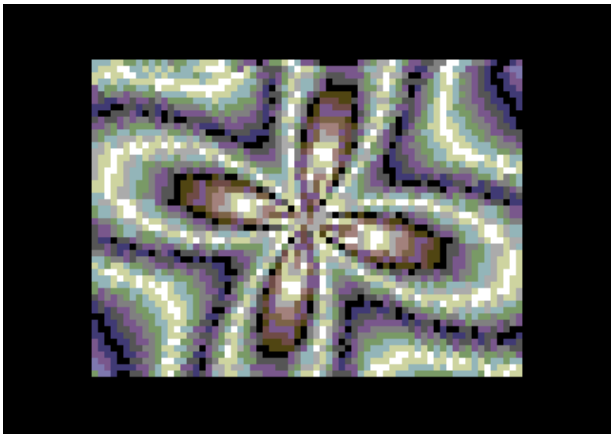
Quiss: Filled vektör'de kullandığım yöntemler şunlardı:

Tüm köşe kordinatları PC üzerinde önceden hesaplanmış ve olabildiğince çok sıkıştırılmış bir veri yapısının içinde saklanıyordu.

Tüm fill işlemleri için XOR-Filler kullanıyordum ("konkav" (içbükey) objeler için bile XOR-Filler kullanıyordum ki normalde bu yalnızca "konveks"(dışbükey) objeler için mümkündür). Köşe koordinatlarının ön hesaplanması oldukça masraflıydı) Aslında, aynı şey C64 assembler kullanarak BSP tree implementasyonuyla realtime olarak da mümkün olabilir (tercihen, Floppy içinde çalışarak [Skate: burada kastettiği şey 1541 ve benzeri modellerde yer alan 2 KB'lık hafıza ve drive'ın işlemcisinin asenkron bir şekilde kullanılması]). Ancak o yıllarda BSP tree'ler hakkında bilgi sahibi değildim.

Ah, ve herşey için XOR-fill kullanılıyordu. Ekranı temizlemek zorunda değildin. Yalnızca çizgilerin yer aldığı tampon belleği silmek yeterli oluyordu.

PC'im üzerinde 3d efektlerin birçok farklı implementasyonu yer almaktaydı. Birçoğu hiçbir zaman su yüzüne çıkmadı çünkü yeterince "sıkı" gözükmüyorlardı (gölgesi düşen bir obje, Z ekseninde clip edilen bir obje vs. de bunlara dahil olmak üzere). Demolarda kullandıklarım hesaplamaları en masraflı olanlar değil, en güzel görünenlerdi (Radio Napalm'daki uzay gemisi ya da shaded küp gibi)



Şekil 17.

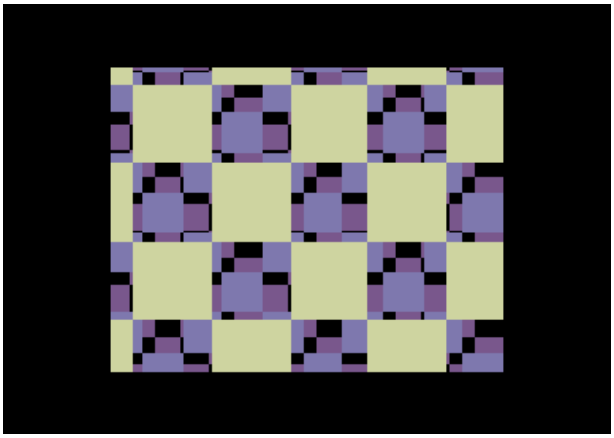
Ah, implementasyonla ilgili son bir nokta: Köşe koorinatlarının hesaplanması için poligonlar PC'de bir painter algoritması kullanılarak bitmap dosyalarına yazılıyorlardı. Daha sonra "ters" XOR kullanılarak çizgiler elde ediliyordu ve köşe koordinatlarını elde etmek için bütün önceden çizilmiş poligon dış hatları geriye doğru takip ediliyordu. Dediğim gibi, o zamanlar henüz BSP treeleri bilmiyordum... ;)

Skate: 1995 yılı için bir diğer çılgın efekt, Gouraud shaded küp. C64 üzerindeki ilk Gouraud shaded vektör bu muydu? O dönemde başka Gouraud shaded vektörler de yapıldığını hatırlıyorum ancak hangisi ilkti emin değilim.

Quiss: Ben de emin değilim. Sanırım Oxyron'dan TTS de bir tane yapmıştı ama hangimizinki önceydi bilemiyorum (Ed: Evet TTS ilkti). Ancak bildiğim birşey varsa O ilk "komplex" (konkav) objeyi bir demoda kullanan adamdır.

Hafızamda kalmış olan bir diğer mevzu ise Gouraud shaded küpün gerçek zamanlı, konkav poligonlar çizebilen, daha üstün bir versiyonunu önceden yapmış olduğumdur. Ancak gerçekten de sıkıcı bir görüntüsü vardı (yalnızca 128x128 boyutundaydı ve biraz yavaştı), dolayısıyla da hiçbir zaman yayınlanmadı. Hem teknik olarak ilginç, hem de iyi görünen bir demo bölümü yazmak her zaman zor olmuştur (Özellikle de demo yarışmalarında oy veren ve çoğunluğu oluşturanların PC'ciler olduğunu göz önünde bulundurursak). Küpün amacına ulaşması beni mutlu etmişti. :) Bir de özellikle Oxyron'dan Graham (zamanının çok sağlam bir programcısıydı [Skate: hala öyle]) beni arayıp, efekti kendisinin de çok beğendiğini söyleyip, tebrik ettiği zaman inanılmaz derecede gurur duymuştum.

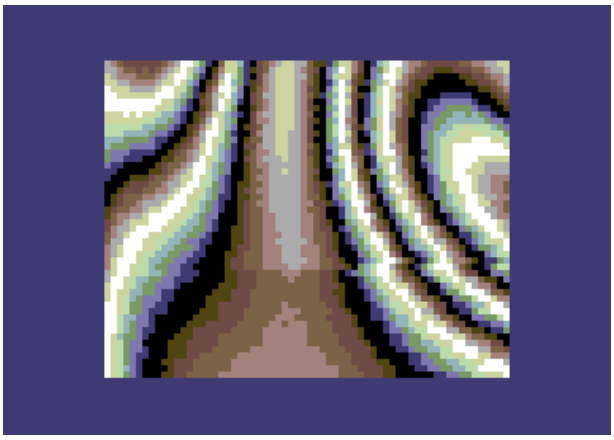
Skate: Ve senin bir diğer buluşun. Sonsuz chessboard zoomer. Bu benim favorilerimden biriydi. Efekti bitirdikten hemen sonra sonucu izlemek tahmin ediyorum çok keyifli olmuştur, haksız mıyım?



Şekil 18.

Quiss: Ben o efekti gece çok geç bir saatte bitirmiştim, dolayısıyla daha fazla izleyemedim çünkü gözlerim yeterince açılamayacak durumdaydı. :) Fikir bir PC 64k introsundan alınmaydı. Ancak PC'dekinden biraz daha gelişmiş bir versiyondu benimki, çünkü izleyicinin PC versiyonunda olduğu gibi çirkin bir clipping oluşturarak içi dolu parçalardan değil de yalnızca boşluklardan ilerleyerek uçmasını sağlamaya ekstra efor sarfedilmişti. (Bu tür bir uçuş bölümü hazırlamak (gerekli veriler C64 üzerinde "siyah ekran" bölümü boyunca hesaplanıyordu) yine sabaha kadar

ayakta kalmama neden olmuştu, çünkü uçuşu hazırlamak chessboard'un kendisini çizmekten çok daha karmaşıktı ki chessboard 'u çizmek yalnızca her raster satırında VIC registerlarını değiştirmekten ibaretti)



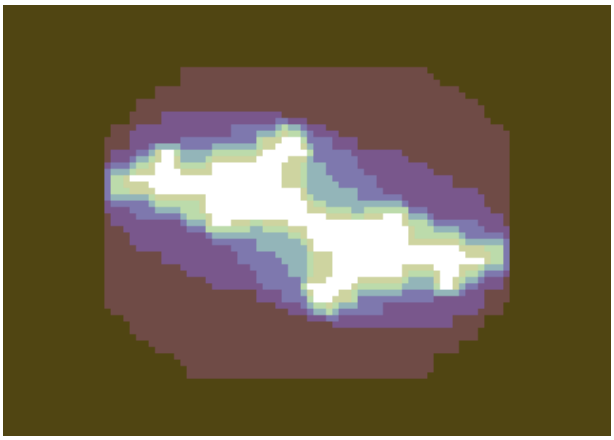
Şekil 19.

Skate: Evet, sıra geldi benim favori Reflex demom olan Mathematica'dan bahsetmeye. Öncelikle auto disk accessing, sid chip autodetection başlangıç olarak teknik açıdan bir harikaydı. Bunun ardından, ilk efekt olarak 5 adet hareket eden çember karşımıza çıkıyor. Daha sonra da bunlar dairelere dönüşüyorlar. Özellikle içi dolu olanları büyük boblardan ibaret de olsalar bu kadar hızlı yapmak oldukça güç. Sanırım zeki bir VIC hilesi kullandınız, öyle değil mi?

Şekil 20.

Quiss: O bölüm Zorc tarafından programlanmıştı. Kendisi o bölüm için dehşet efor harcadı. Eğer doğru hatırlıyorsam, o bölümü bitirmesi aylarını almıştı. Bu arada giriş bölümü de onun tarafından haızrlanmıştı. Bu demoda O'nunla aynı ekipte yer almak gerçekten çok iyi bir şeydi, çünkü herzaman efektlerine maksimum derecede ayar çekerdi.

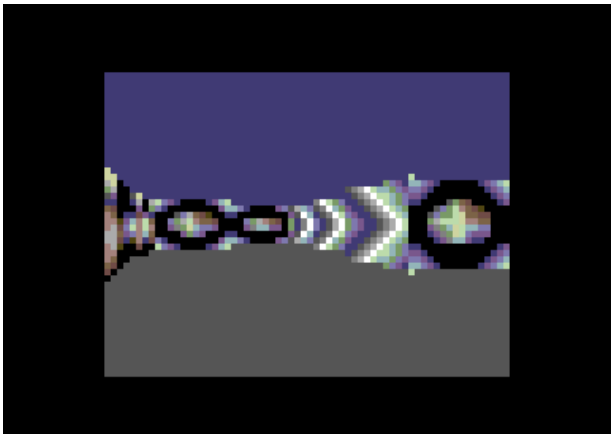
Skate: Gerçek zamanlı hesaplanan bir Julia Fraktal. Mathematica'dan önce ben yalnızca render edilen fraktallar görmüştüm. Bu gerçekten büyük bir yenilikti.



Şekil 21.

Quiss: Şey, efekt alanı yalnızca 20x12 çözünürlüğündeydi (geri kalan pixeller interpole ediliyordu) ama kodlaması zevkli olmuştu. Fraktallar çok büyük bir buluş. Yakın bir zaman önce fraktal sıkıştırmasıyla (fractal compression) ilgili araştırmalar yapmıştım. Buradakiyle aynı şey, yalnızca olayın tersi. Bölüme geri dönecek olursak: Aynı zamanda bu bölüm oluşturduğum hızlı koduma JSR koyduğum ilk bölüm olma özelliğini taşıyor. Bu gerçekten kafamı kurcalamıştı. :)

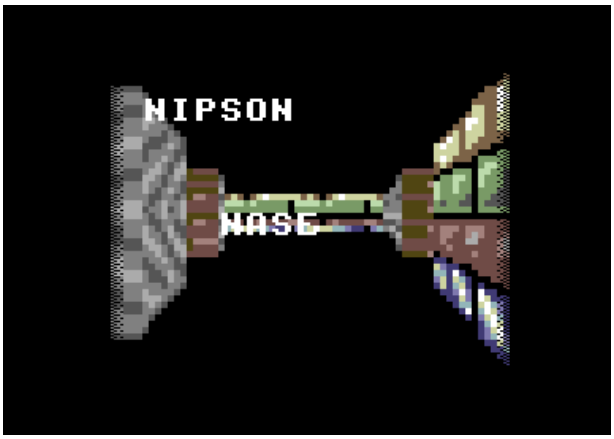
Skate: Ve gelelim birbirinden hızlı iki ray casting engine'e. Yalnızca enginelerin kendisi değil, aynı zamanda sunum biçimleri ve duvarlardaki, 2d spritelardaki efektler de muhteşemdi.



Şekil 22.

Quiss: O engineler Rainer ve benim aramdaki bir yarışmanın ürünüdür. En iyi ray casting engine'i yapmak için kapışmıştık. Bu yüzden Rainer sprite'ı ekledi (ve de greeting yazılarını), ben de duvarlardaki efektleri. Efektin ilk "iterasyonlarında" duvarlardan başka birşey yoktu. Bazen yarışmak sağlıklı sonuçlar verir. :)

Benim raycaster bölümüm tüm demolarımdaki bölümlere kıyasla en çok hafıza kullanan bölüm olmuştu. 64k üzerindeki hemen hemen her byte bu bölüm için kullanılmıştı. Texture verileri, rascasting'in hızlı kodu, kod, müzik, efektlerin hızlı kodları, fontlar, kameranın ilerleyeceği yollar vs. Efektin gösterimi boyunca, sürekli olarak artık kullanılmayacak bölümler silinip, yerlerine sıkıştırılmış yeni veriler açılıyordu vs. Bitirdiğim zaman oldukça mutlu olmuştum.

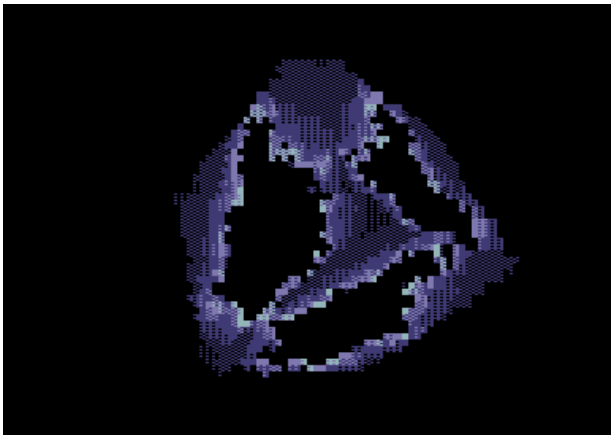


Şekil 23.

Skate: Ben demo kodlarını inceleme konusunda çok tembelimdir. Dolayısıyla doğrudan sana soracağım. Kayan yazı bölümündeki PI sayısı gerçek zamanlı olarak hesaplanıyor muydu yoksa sabit düz yazı mıydı? Sanırım o bir yükleme bölümü değil mi? Yoksa bunca yıl boyunca gözümden kaçan bir özelliği var mıydı? :)

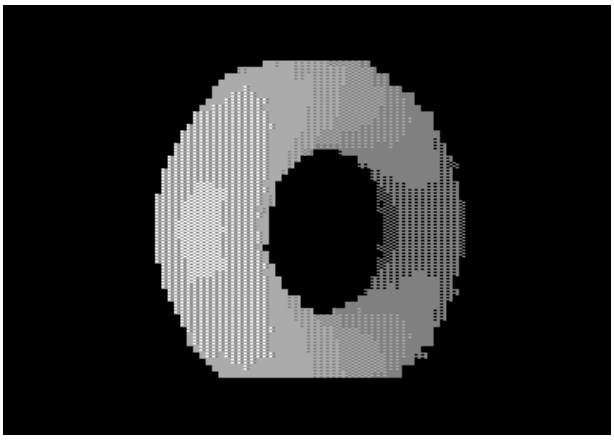
Quiss: O bölüm de Guru tarafından hazırlanmıştı. Sanırım PI sabit olarak girilmişti. Sonuç olarak yalnızca bir yükleme bölümüydü.

Skate: Fraktal texture'lı küp ve shaded torus. Bunlar gerçek zamanlı hesaplanmıyorlardı değil mi? Sanırım video sıkıştırma bölümüydü bu bölümler, doğru mudur?



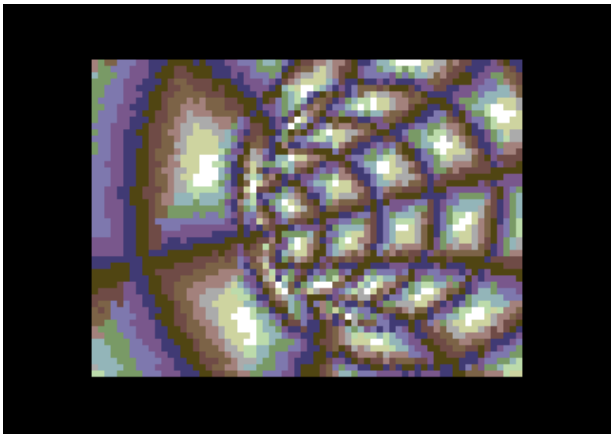
Şekil 24.

Quiss: Hayır, gerçek zamanlı değillerdi. Bunlar birkaç video karesinin ve z eksenine göre zoom efektinin karışımıydı. İlave olarak (torus ve tünel için) renk kayması da kullanılmıştı..



Şekil 25.

Skate: Tamam, artık demolar hakkında bu kadar teknik konuştuğumuz yeter. Seni "Reflection" demosu hakkında dilediğini söylemen konusunda serbest bırakıyorum.



Şekil 26.

Quiss: Demoya kasten sahte krediler eklediğimizi hatırlıyorum ("ssiuq", "fcvp" vs.) çünkü TP C64 demo yarışmasına yalnızca bir demoyla katılmamıza izin vardı. Organizatörler bunu farkettiler ve demoyu yarışma listesinden çıkardılar. (Yine de dev ekranda gösterildi ve bence herşey tamamdı. Aslında oyların geçersiz sayılmasının yanı sıra zaten çok az insan bu demoya oy vermişti)

Skate: Sıra geldi en son releaselerinden biri olan (belki de en son?) "Centric"den konuşmaya. Bu oyun Reflex'in oyun geliştirme bölümü olan Cyance'ın bir prodüksiyonuydu. Yanılmıyorsam ticari bir oyundu. Peki söylesene, macera nasıl gitti? Satmayı başardınız mı?

Quiss: Evet ve hayır. Öncelikle bağımsız olarak satmayı denedik. Posta yoluyla sipariş alıp disketleri yolluyorduk. Bu pek işe yaramadı (16 civarı sipariş aldık). Biz de oyunu 64'er magazine'e sattık.

Sağlam bir kopya önleyici sistem kullandığımızdan ötürü (disketin arka yüzü farklı bir sektör formatı ve dosya sistemi kullanıyordu) 64'er magazine'in oyunu dağıtabilmesi için öncelikle Centric'in crack versiyonunu tedarik etmemiz gerekti. (korumasız bir versiyona ihtiyaçları vardı)

Skate: Neden ve nasıl scene'i bıraktın? Tam olarak ne oldu? C64 scene'ini bırakmanın ardından hiç C64 harici platformları denemedin mi?

Quiss: Şey, biraz PC ile uğraştım, ama hiçbir zaman kendi 3D engine'imi ayağa kaldırıp çalışır bir hale getiremedim. O sıralar okulu da bitirmiştim ve bir yıllığına Alman ordusuna katılmak zorunda kaldım. Bu zaman zarfında bilgisayarda kullanılabilir düzgün birşeyler yapmam mümkün olmadı. Ve hemen sonrasında koleje başladım ki bu zaten dar olan zamanımı iyice azalttı.

Bu yıllar içersinde iyiden iyiye herkesle kontağı kaybettim. Yeniden programlamaya döndüğümde oldukça ilginç şeyler bulunan Open Source camiasına geçtim. Ama asla C64 scene'ini unutmayacağım.

Skate: Aktif scene yıllarını düşünecek olursan, yakın arkadaşların ve düşmanların kimlerdi? Diğer bir deyişler bugün kimlere greetings, kimlere fuckings geçersin?

Quiss: Çok şükür ki hiç düşmanım olmadı, en azından benim bilgim dahilinde.

Arkadaşça greetings yollayacağım kişiler PVCF, Smiling Guru, Zorc, PVT, Felidae, KB, Obsessed Maniacs'dan Luke ve Prince, Smash'den AEG, Oxyron'dan Graham ve RRR, APS'den Seytan ve Sonic ve unuttuğum herkes.

Skate: Biraz da profesyonel iş hayatından söz edelim. "www.quiss.org" URL'inde bir web siten var. Sahte bir fraktal zoomer ile açılıyor ;) O sitede SWFTools, Caiviar, Jissa, sff2jpeg, Athana ve Magpy isminde projeler gördüm. Geliştirmiş olduğun tüm ciddi projeler bunlar mı? Ve birkaç cümleyle bu projeleri bize tarif edebilir misin? Elbette ki tercih edersen istediğin kadar detaya girmekte özgürsün..

Quiss: Fraktal zoomer aslında Oxyron'un Coma Light demolarından birinin bitiş bölümü olan fraktal zoomerın Flash adaptasyonudur. (Gerçekten güzel bir efekt)

SWFTools çok aktif bir mailing list'e sahip oldukça büyük bir projedir. Rainer (Smiling Guru) ve benim tarafımdan başlatılmış bir projedir. Birkaç tane ticari uygulamada da kullanıldı. İnsanlar bu projeyi landmap web portalları oluşturmak, process uygulamaları yayınlamak ve benzeri amaçlarla kullanıyorlar. Şu anda proje o kadar aktif ki, geliştirmelerle ilgili taleplere yetişemiyorum ve C developerlar tutmaya ihtiyaç duymaya başladım (eğer ilgileniyorsanız, kramm@quiss.org adresine e-mail atın).

Caiviar, bir zamanlar bir firma için geliştirdiğim bir uygulamanın Open Source versiyonu. Jissa kolej zamanında geliştirmek zorunda olduğum bir projeden kalma ve Athana ile magpy ise daha büyük bir Open Source projenin ("mediatum") henüz yayınlanmamış parçaları.

Skate: Ben de telekomünikasyon uygulamaları geliştirdiğim için Open Source IVR uygulaması olan Caiviar ilgimi çekti. Sanırım profesyonel yaşantında telekomünikasyon uygulamaları geliştirdin. Peki hangi firmalarla çalıştın?

Quiss: Caiviar'ı, daha doğrusu ticari uygulama olan "mobileX-IVR"ı mobileX (http://www.mobilexag.de/) firması için geliştirdim. Speech Generation ve Voice Recognition konularında Siemens ve Realspeak ile ortaklaşa çalıştık.

Skate: Peki bugünlerde neler yapıyorsun? Artık bir scener olmadığına göre hayat sıkıcı olmalı, değil mi? ;) Tabii ki şaka yapıyorum. Bir karın var mı? Çoluk çocuk?

Quiss: Eşimle birlikte yaşıyorum ve birkaç seneye çocuk yapma planlarımız var. Şu anda Technological University of Munich'de çalışıyorum ve bir taraftan tezimle uğraşıyorum. Boş zamanlarımda çeşitli sporlarla ilgileniyorum (Bisiklet, Snowboard, Kaya tırmanışı vs.) ve Open Source projelerim üstünde çalışıyorum.

Skate: Değerli bir eski scener olarak, yeni başlayanlara tavsiyelerin var mı? İyi bir c64 programcısı olmak için ne yapmalarını önerirsin?

Quiss: Şey, C64 scene'iyle yaklaşık 10 yıldır bağlantım kopuk, şu anda son teknolojinin nerelere geldiğini bilmiyorum. Dolayısıyla vereceğim bilgiler büyük bir oranda zaman aşımına uğramış olacaktır. :) Söyleyebileceğim şey ise şudur ki öncelikle çapraz geliştirme sisteminin ayakta, düzgün çalışır halde olması inanılmaz derecede yardımcı olur. Bunun haricinde, diğer demoları dissassemble etmek ve diğer programcıların efektleri nasıl yaptıklarını incelemek herzaman faydalıdır. Ah, ve diğer platformlardaki (Amiga, PC vs.) demoları seyretmek de yeni fikirler edinmek açısından elbette ki faydalı olacaktır. :)

Skate: Aklında kendine sorulmasını istediğin bir soru var mı? Lütfen kendi kendine sor ve cevapla. :)

Quiss: C64 scene zamanlarında yaptığın en komik şey neydi?

Quiss: Yaptığım en komik şey Guru, KB, Luke ve Sonic ile birlikte (tümü programcı) kalabalık bir trende zaman geçirmek için "bavulunu topla" oyununun C64 assembler versiyonunu oynamaktı. (Kurallar normal "bavulunu topla" oyunundaki gibiydi, yani her oyuncu sırası geldiğinde o ana kadar oluşmuş olan kodu tekrar edip, sonuna bir sıra kod eklemek zorundaydı) Şuna benzer bir şekilde ilerliyordu:

```
Guru:
LDA #$00

KB:
LDA #$00
LDX #$00

Luke:
LDA #$00
LDX #$00
TXA
```

..ve bu şekilde sürer gider. STA $XXXX ile birbirimizin satırlarını

yok edebildiğimiz farkettiğimizde oyun daha da bir zevkli olmaya başladı. PHA ve sonra PLA kullanmak ve benzeri şekillerde devam ediyordu. Deliler gibi gülüyorduk. Muhtemelen trendeki diğer insanlar tamamen aklımızı kaçırdığımızı düşünmeye başlamışlardır. :)

Skate: Scene'e dönüş planların var mı? Var ise hangi platformlarda?

Quiss: Planlanmış birşey yok... şimdilik.

Skate: Teşekkürler Matthias. Seninle bu röportajı yapmak benim için gerçekten büyük bir zevkti. Kendine çok iyi bak. Seninle birgün bir partide tanışmayı ümit ederim. Türkiye'ye her zaman davetlisin. Güzel partilerimiz var burada. Eğer gelirsen benim konuğum olursun.

Quiss: Seninle konuşmak benim için de zevkti.

emir (at) akaydin (nokta) com

Amstrad Köşesi

Türker 'Alcofribas' Gürevin

Amstrad CPC Dünyasına Giriş

Alışılagelmişin dışında bir sistem tanımak isteyenler için fırsat PLAZMA #4 ile ayaklarına kadar geldi nihayet. Hani yıllar öncesinde siz ve mahallenizdeki onlarca çocuk C64'leriniz ile renkli dünyalara dalarken; hatırlar mısınız bilmem ama yeşil ekrana bakmaktan yeşermiş gururlu bir çocuk vardı. Hah işte! O biziz. Kim mi? Kim olacak; daha havalı diye ya kendisinin ısrar ettiği ya da babasının olmuşken iyisi olsun mantığı ile dolduruşa geldiği Amstrad sahipleri. Bahse konu olan yeşilin Hacı veya Dolar yeşili ile uzaktan yakından bir alakası yoktur. İleriki satırlarda neden yeşillendiklerini bulabilirsiniz.

Şekil 1. Muhteşem ikili "Sir Alan Michael Sugar" ve ilk gözağrısı CPC 464

Amstrad kelimesi; firmanın kurucusu olan Alan Michael Sugar kelimelerinin baş harflerinin Trading kelimesi ile ahenkle dansı sonucu(AMS + TRAD) ortaya çıkmıştır. Bu güzel insan daha sonradan, İngiltere'de bolca dağıtılan Sir ünvanlarından birine nail olmuşsa da kendisini Big Brother vari yarışmalarda jüri üyesi olarak veya BBC haberlerinde-belgesellerinde görmek gayet sıradan bir durumdur. Firma, diğerlerinin aksine 90'lardan sonra ortadan kaybolmamıştır. Hatta İngiltere'de Sainsbury'sinden tutun da Tesco, Asda'sına kadar hemen her yerde ses, televizyon, uydu, telefon türünden alet edevatını görmek mümkündür. Misal bizdeki Arçelik-Beko-Vestel ve onların ürün gamı gibi. Ayrıca oldukça sorunsuz ve hızlı olan uydu alıcıları Türkiye'de de yaygın olarak mevcuttur.

8bit dünyasının altın yıllarından biri olan 1984'ün 21 Haziranında İngiltere'de doğan Amstrad CPC (Color Personal Computer), başlangıçta sadece kaset ünitesi ile tümleşik olan 464 modeli olarak piyasaya sürüldü. Zilog firmasının meşhur Z80 işlemcisini kullanan bu 4Mhz'lik canavarın temel çıkış noktası; insanlara kullanması kolay ve ortada kablo yığınlarının dolaşmadığı rahat bir sistem sunmaktı. Yani sokaktaki vatandaş bile kullanabilmeliydi. Amstrad firmasının HI-FI sistemleri üretiminden gelen tecrübesi, bu konuda gerçekten işe yaramıştı. Bu arada AMSOFT(böyle basit esprilere artık gülmüyorsunuzdur umarım) isimli kendi yazılım firması da ayrıca kuruldu.

Ülkemizde tam olarak ne zaman satılmaya başladığını bilmemekle beraber kendi cihazımı 1985 Haziranında(evet bildiniz sınıf geçme hediyesi ile baş ağrısından kurtulma ilacı olarak) almıştım. Biz Sinclair 48K almaya gitmiştik ama babam "hangisi eğitim için daha iyidir" diyince kolumuzun altında bir CPC 464 ile çıkıverdik dükkândan. O yıllarda fiyatlar sanırım şöyle idi: Sinclair 48K(95bin), C64 ekmek kutusu(115bin), CPC 464 Yeşil(145bin). Renkli ekranı hiç hatırlamıyorum. Demek nasıl bir fiyatı varsa, beynim hemen silmiş o lüzumsuz bilgiyi ve yenilere yer açmış.

Şekil 2. AMSTRAD CPC 464

Ürünün Türkiye ithalatçısı olan firma EKAKOMP idi. Bu firma da hala aktif ve oldukça büyük bir elektrik şirketi olarak hayatına devam etmektedir. Lakin CPC serisi cihazları satmanın ve servis vermenin ötesinde, Türkçe kaynak konusunda Teleteknik kadar destekleyici olduğunu söylemek mümkün değil. Katma değerli servislerin hemen hepsi de 3. parti şirketler-kişiler(Memoreks, Körfez Bilgisayar, Mahir Çelikkanat, ElectroMode, Byte Computer-Derya, EGSA-Selçuk, vb) tarafından sağlanmıştır. Tabii o yıllarda 12 yaş civarında olduğumu göz önünde bulundurursak, belki bir ihtimal bir şeyleri atlamış veya farkına varamamış olabi-

lirim. Ama bu bilgiyi düzelten olursa ileriki yazılarımızda telafi ederiz.

Hazır ülkemizdeki durumundan laf açılmışken biraz daha detaya inelim. Elimizde kesin rakamlar olmamakla beraber, bizde de tıpkı dünya genelindeki sıralamada olduğu gibi AMSTRAD satış adetleri birinci sıradaki Commodore ve ikinci sıradaki Sinclair'in ardından üçüncü sırada gelmektedir. Cihaz diğerlerine göre bir miktar pahalı olmasına rağmen daha ciddi bir yazılım(gerçek anlamda iş programları, programlama dilleri, ofis ve inanmayacaksınız ama masaüstü yayıncılık uygulamaları) ve sistem altyapısı(profesyonel klavye, tümleşik depolama, standart monitör) sayesinde başta eğitim kurumları(okullar, kurslar) olmak üzere pek çok kişi tarafından tercih edilmiştir.

Maalesef bu güzel cihaz, monitör olmaksızın satılmıyordu. Her ne kadar Yeşil monokrom (GT64 - GT65) ve Renkli (CTM640 - CTM644) olmak üzere iki alternatif varsa da Renkli ekranı almak için neredeyse bir bilgisayar parası daha vermek gerekiyordu. Haliyle bu da pek çok kişi/aile için kolay değildi. Fakat böyle bir sistemi babasına aldırabilen yiğitler de olmuştur. Tabii monitör işin içine girince fiyat birden yükseliyor ve yeşilden ötesine geçen pek olmuyordu. GT65 ve CTM644 modellerinde diğer iki modelde olmayan, disket sürücü için besleme voltajı çıkışı da bulunmaktadır. İlginç olan, tek başına monitör satışı falan da yoktu. Monitörü bozulan ya da CPC 464'ünü televizyona bağlamak isteyenler için MP1 adı verilen modülatör ünitesi vardı ve bu ünite vasıtasıyla RF çıkış ve sistemi beslemek için gerekli olan 5V sağlanmaktadır. Bunun da fiyatı yine yanlış hatırlamıyorsam 35bin TL civarındaydı. Yukarıdaki rakamlara göre düşünün artık. MP1 sadece 464 modeli ile beraber kullanılabilir. Eğer 664 veya 6128 gibi disket sürücülü bir Amstrad CPC modeline sahipseniz tıpkı GT65 ve CTM 644 model monitörlerde olduğu gibi üzerinde disket sürücü için gerekli olan 12V beslemeyi de sağlayan bir çıkış bulunan MP2 adlı modülatörü kullanmanız gerekmektedir.

Şekil 3. MP2 TV Modülatör

Video çıkışı olarak Motorola 6845 kullanan CPC serisinde temel olarak 3 video modu ve 27 renkten oluşan palet mevcuttur. Her modda farklı sayıda renk kullanılabilmektedir.

- Mode 0: 160×200 piksel ve 16 renk (4 bpp)
- Mode 1: 320×200 piksel ve 4 renk (2 bpp)
- Mode 2: 640×200 piksel ve 2 renk (1 bpp)

DIN5 normundaki RGB video çıkışından doğru bağlantılar yapılarak SCART yoluyla TV çıkışı alınabilir.

Ses entegresi olarak AY-3-8912 kullanan CPC serisi; 7 oktav, 3 kanal ve buna ilave olarak 1 gürültü kanalı sağlar. Dâhili olarak mono bir hoparlörü ve ses seviye kontrolü varsa da, kulaklık çıkışı ile ayrıca stereo ses alınabilmektedir.

Göreceli olarak düşük fiyatı, tümleşik tasarımı, iyi sayılabilecek malzeme kalitesi(MSX cihazlar hariç döneminin pek çok cihazından iyidir), 80 sütun ekranı ve CP/M işletim sistemini de kullanabilmenin getirdiği avantajla; CPC serisi hem ev bilgisayarı pazarı hem de küçük işletmeler için oldukça iyi bir alternatif olmuştur.

CPC 464'ün yakaladığı ciddi satış başarısının rüzgârı aynı yıl CPC 664'ü de beraberinde getirdi. Bu cihazın özelliği ise o dönem için çok ciddi bir adım olan disket sürücü ile tümleşik olması idi. Bunun dışında klavyesinde bir takım değişiklikler vardı ama hafıza veya kayda değer başka bir konuda gelişme olmamıştı. Lakin Amstrad, belki de kişisel tarihinin en önemli hatasını yaparak sürücü markası/modeli seçimi konusunda tercihini Sony'nin 3.5" teknolojisi yerine Hitachi'nin 3" teknolojisinden yana kullandı ve bizler ayvayı yedik. Evet 3" disket sürücüler rivayete göre bir dönem ucuzmuş, ama sonrasında 3.5" lerin yayılması ile sürücü-medya fiyatları hiç düşmedi ve cidden çok yüksek rakamlardan pazarlandılar. Bugün CPC serisi disket ünitesi tümleşik bir makinanın sürücüsünü herhangi bir 3.5" veya 5.25" ile değiştirip, sadece data kablosu üzerinde Ready sinyali ve isteğe bağlı olarak alt-üst kafa / A-B sürücüsü seçimi gibi 3 küçük kısa devre ile tamamen sorunsuz olarak kullanmak mümkün. 3" sürücülerin pin çıkışları 5.25 sürücüler ile tamamen aynıdır. Bu kadar kolay adaptasyon 8bit mikroların(Spectrum 48K, C64, Atari 800XL, MSX) hiçbirinde görmedim. Kayıt medyalarının kapasitesi standart olarak 160K idi ve her iki tarafını da kullanabiliyordunuz(tek kafalı sürücüden dolayı arka tarafını ancak çevirerek). Ayrıca, disketler oldukça sağlam bir yapıya sahipti. Fakat tüm bunlara rağmen 664 oldukça kısa olarak değerlendirebileceğimiz 1 senelik ömründen sonra sahneden çekildi. Bu cihazı ilk ve son kez olarak kendi CPC 464'ümü alırken mağazada görmüştüm. Şimdi harıl harıl arıyorum bir tane.

Şekil 4. AMSTRAD CPC 664

Şekil 5. 3” Disket ve Koruma Kabı

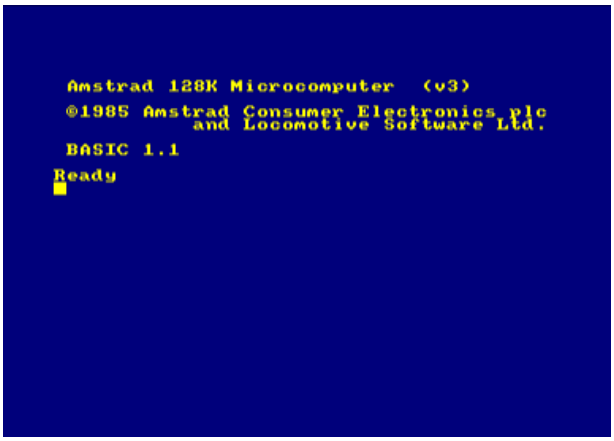
Disket sürücü elbette sadece 664 modeline ait bir konfor değildi. 464 modeli ile beraber disket sürücü kullanmak isterseniz DDI-1 isimli 16K AMSDOS disk kontrol ROMunu’da beraberinde getiren sürücüyü satın almanız gerekmektedir. Bu sisteme ikinci bir sürücü daha bağlamak veya 664-6128 gibi bir modele başka sürücü eklemek isterseniz beraberinde AMSDOS olmayan FD-1 modelini alabilirsiniz.

Şekil 6. DDI-1 ve FD-1 Disket Sürücü

1985 yılı ile beraber 664 yerini çok daha başarılı bir model olan 6128’e bıraktı. Bu yeni cihaz ile beraber toplam 128K hafızaya, daha kompakt ve iyileştirilmiş bir yapıya kavuşmuştuk. Z80 işlemcisinin kısıtlamalarından dolayı 128K hafıza ancak 64K + 64K olarak bank switching metodu ile kullanılabiliyordu.

Şekil 7. AMSTRAD CPC 6128

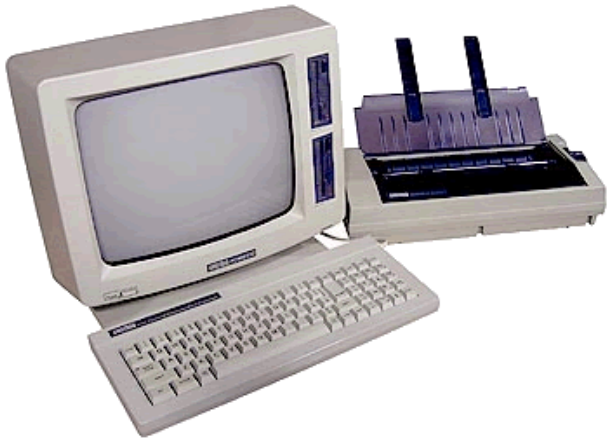
Locomotive Software’in Basic’i ile hazır olarak gelen CPC serisi makinalarda yine ROM’da hazır olarak AmDOS(Amstrad Operating System) ve buna paralel olarak disketten çalışan CP/M 2.2 ve CP/M 3.0 bulunmaktaydı. CP/M işletim sistemi o zaman için programlama dilleri, tablolama yazılımları ve veritabanı gibi ciddi uygulamalar(dBase, Multiplan, Turbo Pascal, C) açısından oldukça zengindi. Tabii BASIC diyince şöyle bir durmak lazım. Amstrad’ın gerçekten çok iyi olduğu bir konu olan Basic gücü, acaba gerçekten cihazın kullanıcı kitlesi için iyi mi olmuştu? Yoksa C64 gibi zayıf bir BASIC ama güçlükler ile uğraşmaya alışmış ve yaratıcı bir kullanıcı kitlesine mi sahip olmak tercih edilirdi? Artık bunu tartışmanın anlamı yoksa da düşünmekte fayda var.



Şekil 8. AMSTRAD CPC Locomotive BASIC Ekran Görüntüsü

Bu arada Amstrad firması; iş dünyasını, özellikle basit sekretarya ve hesap kitap işlerini düşünerek çok daha doğrudan amaca yönelik bir cihaz olan tümleşik ötesi PCW serisini piyasaya sürdü. Bu seri yine Z80 işlemcili, çift 3” disket sürücülü, 256K hafızalı ve tümleşik yeşil ekranı ile 8256 ve 512K hafızası ile 8512 modelleri idi. Genelde yazıcı da dâhil olarak tercih edilen ve Joyce olarak da adlandırılan PCW serisi gerçekten, kullanılmaya başlandığı yerde ihtiyaçlar çok değişmedikçe yıllar

boyu hizmet verebilmekteymiş. Rivayet odur ki bugün İngiltere'de bazı küçük kasaba otellerinde hala kullanılmaktaymış.



Şekil 9. AMSTRAD PCW Serisi

Daha sonra, elbette dönemin popüler yaklaşımı olarak PC serisi cihazlarda Amstrad tarafından üretilmiş. Bunlarda da 1512 ve 1640 gibi hafıza ile orantılı isimler tercih edilmiş. Birkaç ay evvel bunlardan bir tanesini yakında inceleme şansım oldu ve şunu söyleyebilirim ki enfes bir iç-dış tasarım mevcut. Tabii bu noktada şunu hatırlatmakta fayda var ki bizim konumuz PCW ve/veya PC konusu değil.

Olayın burada bittiğini zannediyorsanız gerçekten çok yanılıyorsunuz. 1990 yılında Amstrad oldukça yoğun hissedilen 16bit mikroların baskısı karşısında, 8bitlerin ömrünü biraz daha uzatabilmek için CPC Plus serisini 464+ ve 6128+ modelleri olarak piyasaya sürdü. Bunlar da tıpkı ataları gibi kaset(464+) veya disket(6128+) ile tümleşiklerdi. Gelgelim öylesine önemli yenilikleri vardı ki: kartuş kapısı, AY ses entegresi için DMA, donanımsal scroll, programlanabilir kesmeler, donanımsal 16 adet bağımsız sprite(15 renkli), 4096 renklik palet, 16 bit pazarına(Atari ST, Amiga) hitap eden klavye tasarımı

Şekil 10. AMSTRAD CPC PLUS Serisi

Fakat ne yazık ki bunca güzel özelliğe rağmen, artık çok geçti. Evet, belki Commodore 128 kadar kısır bir şekilde ortada kalmamıştı ama 16bit cihazlar ile mücadele etmesi de mümkün değildi. Uyumluluk konusunda %99 civarında bir performansı olan sistemin tüm özelliklerinden yararlanan 30 küsur adet oyun kartuş olarak yayınlanmıştır. Daha sonradan, bu sistemin GX4000 adıyla bir konsol uyarlaması da piyasaya verilmiştir. Tıpkı Commodore 64GS'ye benzeyen bu cihaz da benzer bir kaderi paylaşmıştır.

Şekil 11. GX 4000 Konsol

Plus serisinden Türkiye'ye girdiğini ne gördüm ne de duydum. Ama sağlam bir kaynaktan öğrendiğime göre bir arkadaş bitpazarından bir tane bulmuş ve onu da Ebay üzerinden yanılmıyorsam İspanya'ya satmış. Kim bilir belki biz de buluruz bir tane…

Dünyada CPC

İspanya, CPC tarihi açısından oldukça farklı bir yere sahip olan 3 ülkeden biridir. CPC 464 olarak bildiğimiz model, hiçbir işlevi olmayan 8K Ram(pcb ye takılmış ama bağlantısı yapılmamış) eklenerek CPC 472 olarak satılmıştır. Buna sebep ise İspanyol yasalarında yer alan "64K'ya kadar olan bilgisayarın" İspanyol alfabesine has harfleri barındırması aksi halde ekstra vergiye tabi olacağı hususudur. CPC bilgisayarlar İspanya'da oldukça tutulmuştur ve oyun alanında pek çok güzel örneğe rastlamak mümkündür.

Almanya'da (Batı) AMSTRAD ismi pek bir şey ifade etmemektedir. Bu ülkede yerel bir firma olan Schneider ismi kullanılmıştır. Cihazlar benzer yapıda olmakla beraber tuş renklerinde ve standart D konektör bağlantılarında farklar vardır. Ayrıca, radyasyon standartlarındaki farklardan dolayı Schneider bilgisayarın içinde koruyucu bir kapak da kullanılmıştır. Burada da oldukça fazla CPC cihaz satılmış ve çok fazla kaynak üretilmiştir.

Doğu Almanya ve Rusya'da CPC modelleri KC compact ismi alt-

ında kopyalanarak üretilmiştir. Bu modellerin dış yapısı tamamen farklı olmasına rağmen %95 civarında bir uyumluluğu verebiliyorlardı. Merkezi işlemci olarak bir Z80 klonu olan U880 ve Amstrad giriş-çıkış işlemcileri klonu olan Z8536 bu cihazların temelini teşkil ediyordu.

Günümüzde CPC

Anavatanı olan İngiltere'de ebay üzerinden epey yüksek hareket oranlarını yakalasa da katma değer üretimi açısından Fransa'nın eline pek kimsenin su dökmesi bugün bile mümkün değildir. CPC scene açısından da referans noktası olan bu ülkede Fransızca olarak pek çok kaynak hala yayınlanmaktadır. Bunlar kimi zaman eski yayınların dijital ortama geçmesi kimi zaman da yepyeni üretimler şeklindedir. Yeri geldikçe bunların hepsine değineceğiz.

Almanya ise geçmişten gelen bağlarının devamı olarak CPC dünyasında etkindir fakat bu noktada maalesef Almancam olmadığı için gerçekçi bir gözlemde bulunmam mümkün değil.

Dikkatimi geçen bir diğer ülke ise en az Fransa kadar aktif olan İspanya'dır. Burada da hem geçmişte hem de günümüzde inanılmaz oranda kaynak üretilmiştir. Arasıra uğradığım bu sitelerde sürekli olarak yenilikler ile karşılaşmak gerçekten etkileyici.

Yunanistan da CPC dünyası açısından aktif gözükmektedir ancak yunanca sayfalar Almancaların da ötesinde bana hiçbir şey ifade etmediği için ne desem anlamsız olur. İtalya, Finlandiya, Norveç, Hollanda ve Avustralya ise CPC açısından artık çok aktif değillerse bile en azından internet ortamında kişisel girişimlerin görüldüğü ülkelerdir.

Genel olarak bakıldığında ise her biri ayrı başlıklarda değerlendirilmesi gereken 2 tane büyük işletim sistemi projesi(Symbos ve FutureOS) ve 1 tane de oldukça ileri düzey bir donanım uygulaması(SymbiFace) bulunmaktadır. Bunlar hakkında ileriki sayılarda detaylı bilgiler bulabileceksiniz.

AMSTRAD CPC Serisi Temel Teknik Özellikleri

	CPC 464	CPC 664	CPC 6128	CPC +464	CPC +6128
Yıl	1984	1985	1985	1990	1990
İşlemci	Z80	Z80	Z80	Z80	Z80
Hız	4MHz	4MHz	4MHz	4MHz	4MHz
RAM	64K (42K)	64K (42K)	128K (64+64)	64K	128K (64+64)
VRAM	16K	16K	16K	16K	16K
ROM	32K	48K	48K	32K	32K
Metin	20x25 (16renk) 40x25 (4renk) 80x25 (2renk)	20x25 (16renk) 40x25 (4renk) 80x25 (2renk)	20x25 (16renk) 40x25 (4renk) 80x25 (2renk)	20x25 (16renk) 40x25 (4renk) 80x25 (2renk)	20x25 (16renk) 40x25 (4renk) 80x25 (2renk)
Grafik	160x200 (16renk) 320x200 (4renk) 640x200 (2renk)	160x200 (16renk) 320x200 (4renk) 640x200 (2renk)	160x200 (16renk) 320x200 (4renk) 640x200 (2renk)	160x200 (16renk) 320x200 (4renk) 640x200 (2renk)	160x200 (16renk) 320x200 (4renk) 640x200 (2renk)
Renk	27-renkli k palet	27-renkli k palet	27-renkli k palet	31(16+15spr) palet4096	31(16+15spr) palet4096
Ses	3 kanal, 7 oktav ve 1 gürültü kanalı	3 kanal, 7 oktav ve 1 gürültü kanalı	3 kanal, 7 oktav ve 1 gürültü kanalı	3 stereo ses, 8 oktav ve 1 gürültü kanalı	3 stereo ses, 8 oktav ve 1 gürültü kanalı
Giriş/ Çıkış	Yazıcı, Genişleme, 1 joystick, Disket sürücü, Monitör, Kulaklık ve Stereo ses çıkışı	Yazıcı, Genişleme, 1 joystick, Disket sürücü, Monitör, Kulaklık ve Stereo ses çıkışı, Harici teyp	Yazıcı, Genişleme, 1 joystick, Disket sürücü, Monitör, Kulaklık ve Stereo ses çıkışı, Harici teyp	Işık Kalemi/ Tabancası, Stereo ses, Centronics, Genişleme, 2 Atari Joystick 1 analojik, kartuş slotu, RGB monitör	Işık Kalemi/ Tabancası, Stereo ses, Centronics, Genişleme, Harici disket, 2 Atari Joystick 1 analojik, Kartuş slotu, RGB monitör
Depolama	Kaset	3" dahili disket sürücü	3" dahili disket sürücü	Kaset	3" dahili disket sürücü
İşletim Sistemi	AMSDOS veya CP/M	AMSDOS veya CP/M	AMSDOS veya CP/M	AMSDOS veya CP/M	AMSDOS veya CP/M
Fiyat	455EU (Mono)	684EU (Mono)	684EU (Mono)		455EU (Mono)

	684EU (Renkli)	913EU (Renkli)	913EU (Renkli)		608EU (Renkli)

AMSTRAD CPC Emülasyonu

İşletim sisteminiz ne olursa olsun, CPC emülasyonu konusunda sıkıntı çekmeniz pek mümkün değildir. Gerçek bir CPC' nin yerini tutmasa da Amiga'dan tutunda cep telefonlarına kadar her tür işletim sistemi için bir CPC emülatörü bulmak mümkündür. Genelde Arnold ve Caprice tüm bu emülasyonlara temel teşkil ediyorsa da pek çok emulatör de kendi özgün yapısı kullanmaktadır.

Konu Windows ortamına gelince, benim şu aralar kullandığım ve favorim olan ise WinApe'dir. Şu an için göze en batan eksiği "sürükle bırak" özelliğinin olmaması. Ürünün web sayfasına www.winape.net [http://www.winape.net/] adresinden ulaşabilirsiniz. Oldukça etkileyici olan temel özellikleri şunlardır:

- Yüksek oranda uyumlu emülasyon
- Windows ortamında en hızlı CPC emülatörlerinde biri
- Dahili Debugger, Disassembler ve Assembler
- DSK/EDSK/ARC ve DSC disket imajlarına destek
- TZX/CDT kaset imajlarına destek
- Otomatik sürüm kontrolü
- Tam SYMBiFACE II emülasyonu (Symbiface CPC'ye donanımsal yetenekler ekler)

Amstrad dosyalarını hemen incelemeye başlamak istiyorsanız sizi çok fazla bekletmeyelim. Emülatörü çalıştırıp nümerik klavyeden 1 tuşuna basın ve sonra şu adımları takip edin:

- File
- Drive A
- Insert Disc Image (dosyayı seçin)
- CAT (bir nevi DIR komutu)
- RUN"DosyaAdı (çift tırnak için Shift ve nümerik olmayan 2'ye basın)
- Enter (tuşuna basın)

Linux/Unix ortamında bir deneyimim olmamasına rağmen CPC forumlarından okuduğum kadarı ile XCPC emülatörü gerçekten başarılıymış. Şu ortamlarda yapılan testleri sorunsuz olarak geçmiş: Gentoo Linux, Mandriva Linux, Fedora Core Linux, SUN Solaris, HP Tru64, SGI IRIX, Mac OS X

Bu emülatörün etkileyici özellikleri ise şunlar:

- Athena GUI (derleme anında seçilen)
- Motif* GUI (derleme anında seçilen)
- Disket imaj desteği (*.dsk)
- Snapshot Hafıza görüntüsü desteği (*.sna)
- Sürükle bırak desteği (*.dsk, *.sna)
- Klavye emülasyonu (QWERTY, AZERTY)
- Joystick emülasyonu (numeric keypad, numlock disabled)
- Neredeyse tam CRTC-6845 / GateArray emülasyonu
- Uyarlanabilir frame-rate

Baktınız bu emülatörler sizi kesmiyor veya gidip dosya indirmeye, zip açmaya üşeniyorsunuz. O halde hemen şu adresi tıklayın:

http://cpc.devilmarkus.de/include.php?path=content/content.php&contentid=46

Tabii unutmadan, her türlü imaj dosyası için size şimdilik ftp://ftp.nvg.unit.no/pub/cpc/ adresini öneriyorum.

AMSTRAD CPC Temel Basic Komutları

İster emülasyon ister gerçek bir CPC' de olsun, programları yükleyip çalıştırabilmek için bazı temel komutların bilinmesi gerekmektedir.

Kaset

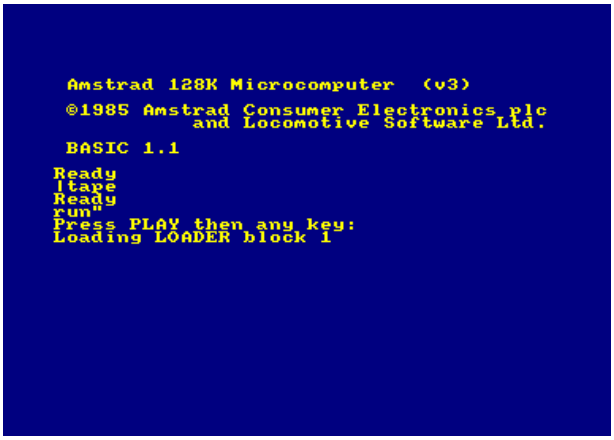
Kasetlerin yüklenmesi için kullanılacak yöntem sistemimizin disket sürücü içerir olmasına göre değişiklik gösterir. Önce kasetinizi teyp ünitesine yerleştirin ve en başa sarın. Eğer disket sürücülü veya disket sürücü bağlı bir sistem kullanıyorsak:

```
|TAPE
```

komutunu vererek makinamızı teyp kullanacak durumu geçiriyoruz. Eğer disket sürücümüz yoksa veya yukarıdaki komutu vermişsek bir sonraki adıma hemen geçebiliriz. Şimdi:

```
RUN"
```

komutunu verirseniz karşınıza "Pres PLAY then any key" mesajı çıkar. Lütfen "any" isminde bir tuş aramayın. Teybinizin PLAY tuşuna ve klavyeden herhangi bir tuşa basın. Komutları doğru vermişseniz karşınızda kısa süre içinde aşağıdakine benzer bir ekran oluşacaktır. Burada Loading <DosyaAdı> block X şeklindeki gösterim işlerin yolunda gittiğini anlatıyor.



Şekil 12. CPC Yükleme Ekranı

Aşağıda, muhtelif hata mesajları ve bunları almanız durumunda yapacaklarınız anlatılmıştır:

“Bad Command” === “|tape” komutunu doğru olarak tuşladığınızdan emin olun.

“Read error a” === Genelde düzelmez. Kaseti başa sarın ve şansınızı tekrar deneyin. Harici teyp kullanıyorsanız ses seviyesini tekrar ayarlayın.

“Read error b === Genelde düzelebilir. Kaseti başa sarın ve şansınızı tekrar deneyin. Harici teyp kullanıyorsanız ses seviyesini tekrar ayarlayın.

Notlar:

"|" işaretini basmak için CPC klavyesinde SHIFT ve @ tuşlarına beraber basınız.

RUN” komutunu kısa yoldan vermek için CPC 464’de CTRL ve nümerik klavyedeki küçük mavi ENTER’a beraber basınız. Aynı komutu CPC 6128’ de vermek için ise CONTROL ve ENTER tuşlarına beraber basınız.

Disket

Disketinizi sürücüye takın, CAT veya |DIR komutunu verin. Bu komut bize dizin listesini verecektir. Bu noktada karşınıza çıkabilecek 3 seçenek var.

```
Drive A: disc missing Retry, Ignore, Cancel?
```

Bu bize disketin sürücüde takılı olmadığını veya sürücünün arızalı olduğunu gösterir.

```
Drive A: read fail Retry, Ignore, Cancel?
```

Bu bize dizinin arızalı olduğunu veya dizin olmadığını gösterir. CPM disketleri ve bazı koruma sistemleri de bunu kullandığından dolayı bir de |CPM komutu ile çalıştırmayı deneyin.

```
Drive A: user 0

DISC.BAS 1K
177K free
```

Dosya ismi olarak “BAS” , “BIN” , “. “ (3 boşluk) görüyorsanız muhtemelen doğru dosyayı tercih ettiniz ve kısaca şöyle yazabilirsiniz:

```
RUN"disc
```

Burada uzun bir şekilde RUN”disc.bas” yazmaya gerek yoktur. “Program load Failed” gibi bir uyarı alırsanız, dizin listesinden başka bir dosya adı seçin ve tekrar deneyin.

CP/M

CP/M disketleri çalıştırmak için klavyeden |CPM komutunu vermeniz yeterlidir. Bunun sonucunda meydana gelmesi muhtemel 3 durum vardır.

```
Drive A: disc missing Retry, Ignore, Cancel?
```

Retry yaptıktan sonra I veya C seçtiğinizde “Failed to load boot sector” mesajı geliyorsa bu bize komutun başarısız olduğunu ve |CPM ile çalıştırılamayacağını gösterir.

```
Failed to load boot sector
```

Bu bize komutun başarısız olduğunu ve |CPM ile çalıştırılamayacağını gösterir.

Ya da son olasılık, program bir mesaj vermeden yüklenir ve çalışır. |CPM komutu başarılı olmuştur.

Şimdilik bu kadar, bana ulaşmak isterseniz aşağıdaki adresi kullanabilirsiniz.

8bitmicro (at) gmail (nokta) com

Rakı, Balık, Spectrum

Arda Ö. 'Ref' Erdikmen

Şekil 1.

Simsiyah ve oval hatlara sahip bir gövde üzerinde, koyu turkuaz renkte kauçuk tuşlar kutuyu açtığımda gördüğüm ilk şeydi. Bir hayal kırıklığıyla kutudan çıkardığım alet babamın hesap makinesinden biraz büyük, biraz inceydi. Büyük bir dikkatle sehpanın önüne koydum ve kutuya geri döndüm. Ne bir oyun kolu, ne de kartuş vardı. Sadece kendi gibi kapkara bir teyp kasedi…

Ben Arda Ö. Erdikmen, 1983 yılında ZX Spectrumla tanışmam bu şekilde olmuştu. Benim kuşağımdaki tüm erkek çocuklarının buna benzer hikayeleri vardır. İşte bu hikayeler bizim için eski bilgisayarlarımızı özel yapar. Horace ya da Last Ninja yüklenirken ekran yanında verilmiş pozlar, babalarımızın çemberleri ya da ilk bisikletleri ile çektirilmiş pozlara benzer. Bilgisayarın başına geçmek için ailemizle yaptığımız kavgalar ve antlaşmalar, upuzun listeden sadece birkaç oyun seçmek zorunda olmanın baskısı içimize işlemiştir.

Bu sayıda size zx spectrum'a ait anılardan bahsedeceğim. Böylelikle retro kültüre yabancı olanlar Spectrum'un kokusunu biraz alabilirler umarım. Spectrum'u spectrum yapan başlıklara değinirken teknik gargaradan kaçınmaya çalışacağım.

İşte zx spectrum'un "top 10"i:

Chuntey: Tüm Spectrumcular kesinlikle bu konuda hemfikir olacaklardır: teypten yükleme yaparken her an, herhangi bir sebep olmadan yüklemeniz yarıda kalabilir. Bu sebepten yükleme sırasında bilgisayarın "Aura"sı bozulmamalı, yükleme rahatsız edilmemelidir. Spectrumcuların bu auraya verdikleri isim "Chuntey"dir. En sevdiğiniz oyun yüklenirken annenizin kapıdan içeriye çikolatalı ekmek ve çay ile girmesi chuntey üzerinde ağır bir rezonans oluşturup "R Tape loading error" hatasına sebebiyet verebilir.

Şekil 2.

Spectrum: Zx Spectrum aslında sinclair'in ZX81 serisi bilgisayarlarının devamı. Fakat bu modelden önce mono olan görüntü çıkışına fazladan 15 gökkuşağı rengi eklendiği için buna "Spectrum" kod adı verilmiş.

Yükleme hızı: Standart yükleme hızında 48kb bilgiyi yaklaşık dörtbuçuk dakikada teypten yükleyebilir.

Joffa: Jonathan Smith, nam-ı diğer Joffa spectrum'un en ünlü programcılarından biridir. Kendisi zamanında ocean, imagine, specialFX gibi spectrum'a dağıtım yapam büyük firmalarla çalışmış olmasının yanında ünü, kullandığı çok hızlı titreşimsiz ekran tazeleme kodundan gelir. Yani spectrum'a tüm ekranı bir piksellik keskinlikle kaydırma yeteneğini sağlayan kodu ilk yazan adamdır (Joffa stack'i ekranın başlangıç adresine ayarlayıp veriyi stack'e push ederek çizimi gerçekleştiriyordu). Bunun yanında joffa her oyununda kendi adını çok acaip yöntemlerle (ters, ayna, kendi ismiyle dalga geçerek vs.) yazarak bir stil oluşturmuş, farkında olmadan adının duyulmasını da sağlamıştır.

6031769: Matthew Smith'in sürücü belgesinin numarası, spectrumcu kuşağın ezbere bildiği bir rakam olmuştur. (bu numara ile manic miner'da cheat yapabiliyorsunuz)

"Kesinlikle stonkers değil": Spectrum'da onbinlerce oyun var, ve elbette birileri size daha önce oynadığı bir oyunun adını soracaktır. İşte burada yine spectrumculara özgü bir cevapla işe başlamak gerekiyor: "Kesinlikle stonkers değil!". Aslında stonkers, 1983'de çıkmış, gerçek zamanlı bir strateji oyunu (en azından denemişler). Hem stonker'in bir argo karşılığı olması, hem de oyunun pek birşeye benzememesi bu cümleyi zaman içinde spectrum popüler kültürü içerisine sokmuş.

Chaos Constructions: Spectrum camiasının alternatif yüzünü gösteren, doğu bloğu ülkeleri programcılarını bir araya getiren en önemli etkinlik, St.Petersburg'da düzenlenen demopartisi. 11 yıldır düzenlenen festivalin yarışma bölümünde 7 ayrı zx spectrum kategorisi bulunuyor ve bu partide her yıl yüklü miktarda ürün çıkıyor.

Sertifika: http://www.worldofspectrum.org/testzx.cgi adresinde bulunan ne kadar iyi bir spectrum kullanıcısı olduğunuzu ölçerek

size bir sertifika veren bir test var. Soru bankasında, farklı kategorilerde sorular var ve bunlar size rastgele soruluyor. Spectrum Okulunun geçme notu ise 70. Sınavı almadan önce iyi çalışın, göründüğü kadar kolay değil.

WorldOfSpectrum: Martijn Van Heide'nin oluşturtuğu en geniş ve tamamen legal zx spectrum yazılım veritabanı. Sitenin yöneticisi ve arkadaşları, tesbit edebildikleri her yazılım evi ve yazara ulaşarak, onlardan yazılı izin belgeleri alıyorlar ve bu izinlerden sonra programları arşive ekliyorlar. Spectrum'un son sahibi Amstrad, "world of spectrum"un bağlantı kurmasının ardından, tüm spectrum ürünü yazılımlarını public domain haline getirdi, bunun sonucunda emülasyon programları ve alternatif rom sürümleri hızla çoğaldı.

Şekil 3.

Sir Clive Sinclair: Sinclair bilgisayarlarının yaratıcısı. Bir dönem zx spectrum satışları o kadar iyi gidiyordu ki, bir tv röportajında zx spectrum için neden disket sürücü yerine manyetik bant esaslı yükleme sistemi "microdrive"ı sattıklarını soran muhabir bayana, "Biz bu eleştiriyi yapan tüm şirketlerden (commodore ve tüm msx üreticilerini kastederek) daha çok ürün satıyoruz, markette söz sahibi olduğumuza inanıyor ve kendimize özel seçimlerimizi yapabileceğimizi biliyoruz" diyecek kadar burnu kalkmıştı.

…Halbuki benim anneme sipariş verdiğim alet bir VCS2600 değil miydi? Ahhh, ne kadar hatalıymışım. İyi ki annem renk benzerliğine kanıp bir zx spectrum almış.

Emulatöre AmigaOS Kurun

Ahmet 'Datura' Moldibi

Amiga işletim sistemini kullanmak için iki yol vardır, Amiga'ya sahipsinizdir ve işletim sisteminizi sabitdiskinize kurarsınız ve mutlu olursunuz. Ya da Amiganız yoktur (varsa da konfigurasyonu düşüktür) ve emulasyon yazılımları sayesinde sanal bir Amiga edinir onun üzerine Amiga OS kurarak mutlu olursunuz. Görüldüğü gibi Amiga kullanıp da mutlu olmanın yolu Amiga OS'u kurmaktan geçer:) Peki emulasyonla eski oyunlarımızı oynamakla beraber bir de Amiga işletim sistemi kurup kullanabiliyorsak o halde bunu neden yapmayalım? Üstelik de 10 dakikamızı alacaksa...

Bilirsiniz Amiga emulasyonu denilince akla ilk gelen isim WinUAE'dir. PC donanımlarının AmigaOS'u yeteri kadar iyi emule edecek düzeyde hızlı olmadığı dönemlerin geride kalması, bu süreçte WinUAE'nin de etkili bir şekilde geliştirilmeye devam etmesiyle artık gerçek Amiga'ya çok yakın sanal sistemler oluşturmak mümkün hale geldi. Biz kullanıcılara da bunu değerlendirmek için çeşitli imkanlar doğdu.

Bu yazımız tabii ki bir "WinUAE rehberi" olmayacak çünkü bunun için gerekli kaynaklara kolayca erişmek mümkün. Burada asıl amacımız gereksiz detaylara girmeden bir sistem kurmak olacak. Çünkü birçok kullanıcı Amiga'yı donanımsal olarak çok fazla tanımayıp bazı ayarları yanlış yapmakta, sonuçta yavaş ya da problemli bir emulasyonla beraber bu denemelerden vazgeçmektedir. Ya da sanal Amigası çalışsa da işletim sistemi kurma konusunda nereden başlayacağını bilememektedir.

Gerekli birçok yazılım Internet'te mevcut olduğundan bunları bulmak güç değil (mesela yıllardır www.aminet.net bu konuda en önemli kaynaklardan birisidir). Yazılımları indirmek ve kurmak, sistemi kullanışlı hale getirmek için gerekli ince ayarlarla uğraşmak ise vakit isteyen bir iş. Peki bu noktada ne yapılabilir?

Bizim yerimize gerekli birçok yazılımı sistemine kurmuş ve stabilite testlerini yapmış, daha sonra da bunu başka kullanmak isteyenlerin hizmetine sunmuş birileri olsa ne iyi olurdu. Hiç uğraşmadan hazır kurulu bir sistem elde ederdik. Dilediğimiz zaman sistemi tekrar sil baştan kurmak birkaç dakikalık iş olur, ilk haline getirmek ise birkaç saniyemize malolurdu. (Kurulu sistemimizi bir başka klasöre yedek aldığımızda dilediğimizde oradan geri kopyalayabiliriz)

Sonunda birileri kullanıcıların bu isteklerini duymuş olacak ki, tam da bahsettiğimiz gibi bir çözümü ücretsiz bir şekilde herkesin kullanımına sunmuş. İşte bunlar, bu sayıdaki yazımızın da konusu olan AmigaSYS ve belki bir başka sayıda konu edineceğimiz AmiKit.

Hayat kolaylaştıran iki yazılım paketi: AmigaSYS ve AmiKit

İki farklı ekip tarafından geliştirilmekte olan ve oldukça da sık güncellenen Amiga SYS ve AmiKit, Amiga OS'u emulasyonla kullanmak isteyen kullanıcılar için bahsettiğimiz bu önemli ihtiyacı gidermeye çalışan değerli iki proje. Her iki proje de zaman kaybını en aza indirerek işlevsel ve güncel bir Amiga sistemi kurmayı sağlıyor. Aslında her ikisini de kurulmuş ve kullanıma hazır yüzlerce yazılımı içerisinde barındıran paketler olarak düşünebilirsiniz.

Bu paketleri Windows'ta, GNU/Linux, BeOS, AROS, Amiga OS4, MorphOS gibi işletim sistemleri üzerindeki UAE emulasyon yazılımı üzerinde sanal olarak çalışan Amiga sistemimize kurup kullanabiliyoruz.

Bu yazıda örnek olarak AmigaSYS paketini GNU/Linux bir sistem üzerinde kurarak, ardından Amiga OS 3.9 yükseltmesini yapacağız.

E-UAE ve Amiga Emulasyonu

E-UAE yazılımı WinUAE kaynak kodlarından yola çıkılarak, GNU/Linux, MacOS X, BeOS, AROS, MorphOS ve AmigaOS4 sistemleri gibi alternatif birçok sistemde açık kaynaklı bir Amiga emulatosyonu sağlamakta. WinUAE'nin kendi gelişimini sürdürerek kazanmış olduğu özellikleri bu işletim sistemlerine de taşıyan E-UAE'nin Windows harici sistemlerde -henüz çok yolu olsa da- büyük bir ihtiyacı doldurduğu muhakkak. Burada anlattığım konuları GNU/Linux haricinde E-UAE kullanabildiğiniz yukarıda saydığım bütün sistemlerde deneyebilirsiniz.

Amiga OS Kurulumu

Bize Gerekenler

1. E-UAE Amiga Emulasyon yazılımı(www.rcdrummond.net/uae/): x86 tabanlı bir pc kullandığım için "e-uae_0.8.29-WIP4_linux-i586_sdl.tar.bz2" isimli dosyayı indirdik. Siz web sitesinden her zaman en güncel sürümü indirebilirsiniz.

2. AmigaSYS 3 Plus E-UAE, hayatımızı kolaylaştıran yazılım paketimiz (amigasys.extra.hu/downloadamigasys.html) : AmigaSYS'nin E-UAE için hazırlanmış bu önkurulu (preinstalled) paketini web adresinden indirebilirsiniz. Yukarıda sıraladığım Windows harici bütün sistemler için Amiga OS kurarken bu paketten yararlanacağız. Şu anda GNU/Linux üzerinde kuracağımızdan, AmigaSYS 3 Plus E-UAE HDF başlıklı bağlantıdaki "AmigaSYS3PlusEUAE.zip" isimli dosyayı indirdik)

3. Amiga Kickstart 3.1 ROM dosyaları (http://www.amigaforever.com/): Elinizde mevcut değilse, belirttiğim web adresinden Amiga Forever paketini satın

alarak lisanslı Amiga ROM dosyalarını edinmeniz mümkün.

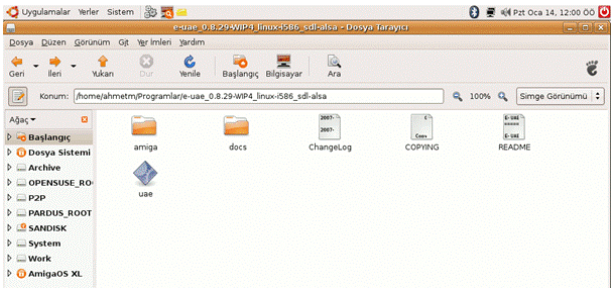
4. Amiga Workbench disketleri, yani ADF formatındaki disk imajları: Kurulum sırasında bize sadece workbench ve extras disketleri gerekecek. Çünkü öncelikle sistemimizi OS 3.1 kurulu hale getirmemiz gerekiyor.

5. Amiga OS 3.9 ya da Amiga OsXL-Amithlon CD'si (http://www.amigakit.com veya http://www.vesalia.de)

Evet elimizdekiler hazırsa artık başlayabiliriz. Her ne kadar burada Ubuntu dağıtımı üzerinde anlatacak olsak da bu işlemleri herhangi bir GNU/Linux dağıtımı üzerinde de tamamen aynı şekilde gerçekleştirebilirsiniz. Çünkü dağıtıma özel farklılıklar söz konusu değil.

Kuruluma hazırlık

İlk iş olarak kendi sistemimdeki home dizinime (veya dilediğiniz farklı bir yere klasör açarak onun içerisine -mesela ben "Programlar" isimli bir klasör içerisine açtım-) daha önce indirmiş olduğum AmigaSYS3PlusEUAE.zip isimli dosyayı açıyorum.



Şekil 1.

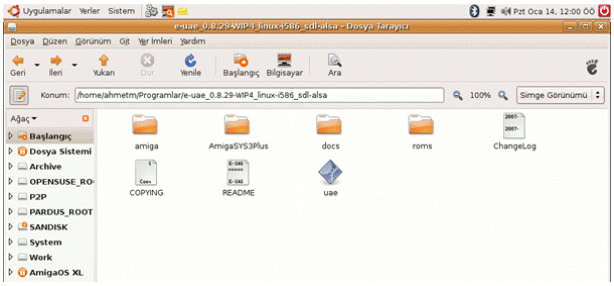
Artık elimde "e-uae_0.8.29-WIP4_linux-i586_sdl-alsa" isimli bir klasör ve içerisinde daha önce derlenmiş ve çalışmaya hazır durumda olan E-UAE yazılımı mevcut hazır durumda.

Şimdi E-UAE program dizininin içerisinde bize gerekecek diğer klasörleri ve dosyaları hazırlayalım:

1. "roms" ve "AmigaSYS3Plus" isimlerini verdiğimiz iki klasör oluşturduk.

2. AmigaSYS3PlusEUAE.zip dosyasını yeni oluşturduğumuz "AmigaSYS3Plus" klasörünün içine açıyoruz.

3. Amiga Kickstart 3.1 ROM dosyamızı "roms" klasörünün içerisine kopyalıyoruz.



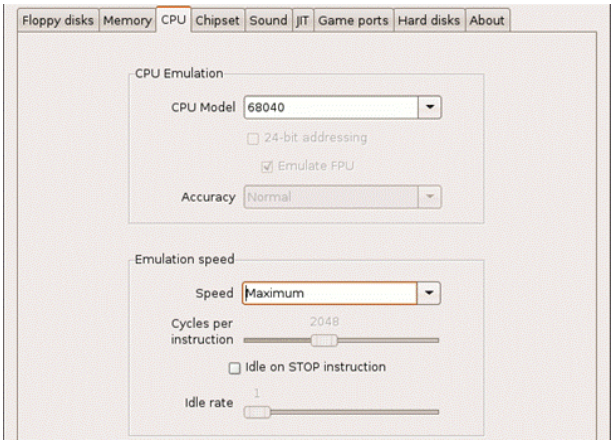
Şekil 2.

Evet son durumda artık E-UAE'yi çalıştırmaya hazır hale geldik. Şimdi bir sonraki adıma geçebiliriz.

E-UAE'de gerekli ayarlar

Bu yazının sonunda Amiga modellerini donanım yönünden iyi tanımayan kullanıcılar için pratik bir rehber olacak bazı ek bilgileri bir tablo halinde bulacaksınız. Bu bilgileri bilmek size emule etmek istediğiniz Amiga modeli için gerekli ayarlamaları en iyi biçimde yapma imkanı verecektir.

Hemen verimli bir emulasyon için ayarları yapalım. E-UAE emulatorumuzu uae simgesine çift tıklayarak çalıştırdığımızda programın penceresi karşımıza gelecektir. Karşımıza gelen program penceresi oldukça sade bir arayüz sunmakta. Bir File menusu var ve içerisinde "save config" başlığı bulunuyor. Diğer bütün arayüz öğeleri pencerenin üzerinde sıralanan panellerde yer alıyor. Şimdi sırasıyla bize gerekli olan ayarlamaların yer aldığı panelleri dolaşalım ve gerekli yerleri düzenleyelim.



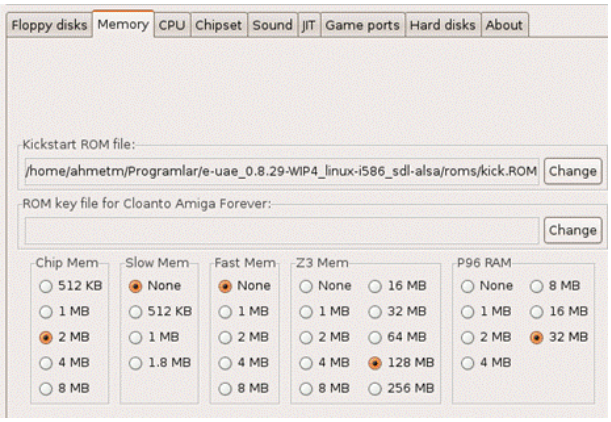
Şekil 3.

Öncelikle CPU panelini tıklayıp CPU emulation bölümünde işlemcimizi 68040 olarak seçelim. Daha altta yer alan Emulation speed bölümünde ise Speed Maximum seçelim. Bu yaptığımız ayarlar ile artık Memory panelinde ilk başta seçmemize izin verilmeyen memory ayarları erişimimize açılmış olacaktır. (Çünkü varsayılan ayarlar Amiga 500 'e göre geliyor)

Şimdi Memory paneline geçelim ve Kickstart ROM File kısmından Change düğmesini tıklayarak "roms" klasörümüze kopyalamış olduğumuz kickstart 3.1 rom dosyamızı bulup seçelim. (Benim sistemimde kick.ROM isimli dosya)

Daha sonra da alt tarafta yer alan memory seçeneklerinde;

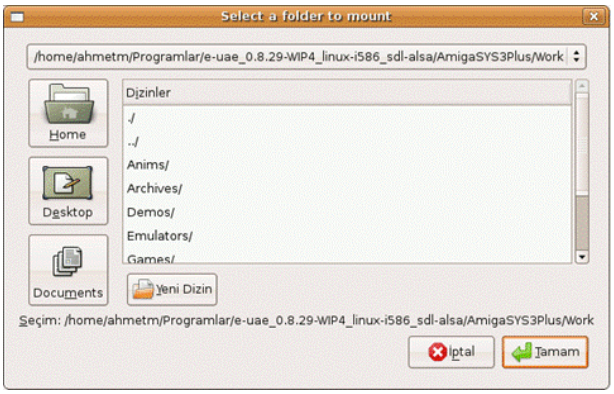
1. Chip Mem: 2Mb
2. Slow Mem: none
3. Fast Mem: none
4. Z3 Mem: 128Mb
5. P96 Ram: 32Mb



Şekil 4.

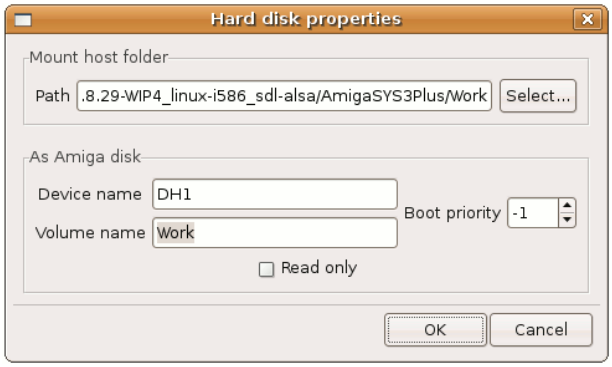
olarak ayarlayalım. (Buradaki amacımız Amiga OS 3.9 sistemini performanslı çalıştırabilecek 68040 işlemciye yükseltilmiş bir Amiga1200 elde etmek.)

Şimdi Chipset paneline geçebiliriz, burada da Chipset Model olarak AGA seçtik ve böylece bizim için önemli ve gerekli ayarlamaları E-UAE'nin arayüzü üzerinden yapmış olduk. Tek bir şey kaldı o da Hard disks panelinde yer almakta olan disk ayarları.



Şekil 5.

Şimdi "Hard disks" paneline geçiyoruz. Alttaki "Add ..." yazan düğmeye tıklıyoruz ve buradan karşımıza gelen pencerede Path yazan kısımda Select'i tıklayıp dosya seçim penceresine giriyoruz, Work partition'ı olarak kullanacağımız (Amiga sistemimizdeki ikinci disk bölümü) klasörü seçiyoruz. Tabiiki bunun için AmigaSYS3Plus klasörüne giriyoruz, Work klasörüne giriyoruz ve "Tamam" tıklıyoruz..



Şekil 6.

Şimdi de disk bölümümüze gerekli isimleri verelim, bunun için "As Amiga Disk" yazılı bölümde "Device Name" kısmına DH1 yazınız. (Amigada harddisk device genellikle dh0, dh1, dh2 şeklinde veya hd0, hd1, hd2 şeklinde isimler alır. Burada Work disk bölümünü DH1 olarak yani ikinci disk bölümümüz olarak belirlemiş olduk). Bundan sonra da "Volume Name" kısmına Work yazıyoruz. (Böylece DH1 device ismine sistem içerisinde kullanırken kolaylık olsun diye Work şeklinde bir isim atamış olduk)

Son olarak da Boot Priority kısmını -1 olarak seçiyoruz (Böylece boot etme önceliğini daha yüksek öncelik değeri vereceğimiz system disk bölümüne bırakacağız) Burada da işimiz bitti "OK" tıkladığımızda Hard disks panelinde disk bölümümüzü görebilirsiniz.

Artık File menusunden "save config" seçerek ayarlamızı kaydedebiliriz. Dikkat ettiyseniz System (DH0) disk bölümümüzü henüz ayarlamadık. Çünkü burada küçük bir problem bizi bekliyor. Ayar panelinde sadece bir "klasörü" harddisk olarak gösterme imkanı bize sunulmuş. Oysa bizim AmigaSYS paketinin içinden System isimli klasör içerisinde bir HDF (harddisk imaj dosyası) ve bir de Work isimli klasör çıkmıştı. İkincil disk bölümü olarak Work (DH1) dizinini az önce ayarladık. Peki System disk bölümünü bu durumda nasıl ayarlayacağız? Bu işi bir text editor ile E-UAE konfigurasyon dosyasını açıp elle editleyerek yapmamız gerekmekte.

Şimdi bu kısmı yapmak için E-UAE programını "Quit" düğmesine tıklayarak kapatalım (öncesinde ayarlarımızı File menusunden "save config" tıklayarak kaydetmeyi unutmamalıyız) ve bir terminal penceresi açalım. Halihazırda home dizinimizde olmamız lazım, değilsek home dizinimize geçelim. Burada ls -al komutunu vererek dizinde bulunan dosyalari listeleyelim. Az önce e-uae ayarlamızı kaydettiğimizde burada bir konfigurasyon dosyası oluşmuş olmalı. Bu dosyanın ismi .uaerc şeklinde görünüyor olmalı. Şimdi bu dosyayı edit ederek gerekli disk bölümü

bilgisini elle gireceğiz. Ayrıca burada bir işimiz daha olacak o da Amiga OS'u boot ettiğimizde Linux üzerinden eriştiğimiz CD veya DVD sürücümüze Amiga'dan da erişebilmek için yine bir ayarlama yapacağız. (Bu önemli çünkü AmigaOS 3.9 kurmak istiyorsak sisteme CD okutabilmemiz gerekiyor)

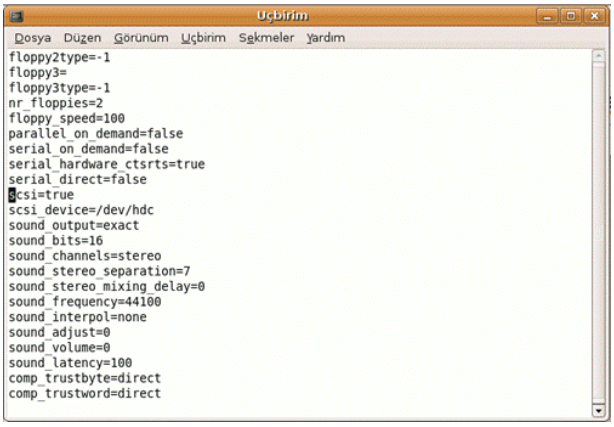
Konsolda (terminal penceremiz) vi .uaerc yazıp enter'a bastığımızda dosyanın içeriği karşımıza gelecektir. (shell konsolunda vi editorunu kullanmaya yabancıysanız http://users.tkk.fi/~thyle/vi_opas.html adresindeki hızlı kullanım bilgileri işinize yarayacaktır) Şimdi edit edeceğimiz kısımlara sırayla gidelim.

Dilerseniz öncelikle CD/DVD sürücü için gerekli ayarları ekleyelim. Bunun için Hardware Information programını çalıştırmamızda yarar var, CD/DVD sürücümüz Linux'ta hangi cihaz olarak sisteme tanıtılmış bunu öğrenmeliyiz. (Ubuntu dağıtımında "Hardware Information" programını [HAL Device Manager] "Sistem > Yeğlenenler" menusu altında bulabilirsiniz. Open Suse kullananlar da Yast çalıştırarak sistem bilgilerini görüntüleyebilirler.)

Şekil 7.

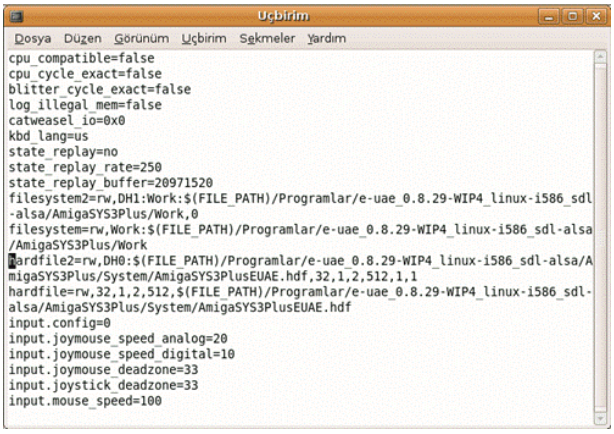
DVD sürücümüzü Devices listesi üzerinden seçip Gelişmiş panele geçtiğimizde cihazımın linux dosya sisteminde nasıl tanıtılmış olduğunu görüyoruz. Örneğin benim sistemimde /dev/hdc olarak tanıtılmış durumda. (Sizde ise /dev/hdd ya da /dev/hdb gibi farklı bir bağ almış olabilir) O halde gerekli bilgiyi aldık şimdi tekrar terminal penceremize dönerek kaldığımız yerden devam edelim.

E-UAE konfigurasyon dosyası içerisinde scsi kelimesiyle başlayan iki satır bulunmakta, bu satırları buluyoruz ve "scsi=" yazan yerin yanına true hemen alt satırındaki "scsi_device=" yazan yerin yanına da /dev/hdc (benim sistemimdeki cd/dvd-rom sürücüm bu bağlantıyı kullandığı için, sizde farklı olabilir) yazıyoruz. Böylece CD/DVD-Rom sürücümüze Amiga içerinden erişebilmek için gerekli olan ayarlamayı yapmış olduk. Şimdi System partition için gerekli ayarı girelim.



Şekil 8.

Konfigurasyon dosyamızda daha alt satırlarda bir yerde "filesystem2" ve "filesystem" ile başlayan birer satır göreceksiniz. İşte bu iki satır aslında E-UAE'yi çalıştırıp DH1 device ayarını yapıp kaydetmemizden sonra konfigurasyon dosyamıza yerleşti. Şimdi bu satırların hemen altına yeni bir satır açıyoruz. (Ed: Bu satıra ve ardından gelene girilen bilgiyi biz aşağıda 4 ve 3 satır olarak göstermiş olsak da siz bu 4 satırı arada hiç boşluk bırakmadan birbiri ardına tıpkı Şekil 9'da görüldüğü gibi girmelisiniz)



Şekil 9.

```
hardfile2=rw,DH0:$(FILE_PATH)/Programlar/
e-uae_0.8.29-WIP4_linux-i586_sdl-alsa/
AmigaSYS3Plus/System/AmigaSYS3PlusEUAE.hdf,
32,1,2,512,1,1
```

yazıyoruz ve bir alt satıra da aynı şekilde;

```
hardfile=rw,32,1,2,512,$(FILE_PATH)/Programlar/
e-uae_0.8.29-WIP4_linux-i586_sdl-alsa/
AmigaSYS3Plus/System/AmigaSYS3PlusEUAE.hdf
```

satırını ekliyoruz. (Dilerseniz kopyala-yapıştır da yapabilirsiniz)

Burada dikkat edilmesi gereken nokta gerekli dosya yollarının benim sistemime göre olduğudur. Yani home dizinimde "Programlar" dizini oluşturarak içerisine e-uae dizinini açtığımdan dolayı yukarıdaki yolda "/Programlar/e-uae_0.8.29-WIP4_linux-i586_sdl-alsa" şeklinde gerekli satırı girdim. Siz kendi sisteminize göre bu kısmı düzenlemelisiniz.

Artık konfigurasyon dosyasında da işimiz bittiğine göre (konsolda vi ile editlediyseniz escape tuşu ile editleme modundan çıkalım ve : [iki nokta üstüste] işareti yazarak komut satırına çıkıp x! yazıp enter'a basarak kaydedebilirsiniz) dosyayı kaydedelim.

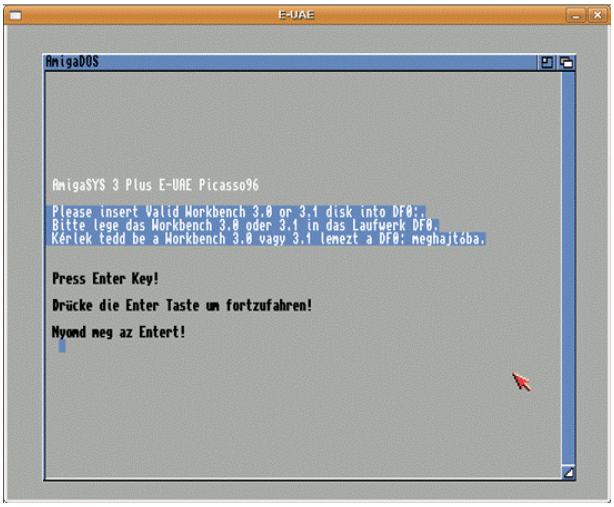
Artık bütün ayarlamalarımız tamam olduğuna göre Amiga OS kurulumuna geçebiliriz. AmigaSYS paketi bize hazır bir OS3.1 sistemini sunmakta olduğundan, biz emulatorümüzü başlattığımızda bizden yalnızca 3.1 workbench disketleri isteyecektir.

Amiga OS 3.1 Kurulumu

E-UAE simgesini çift tıklayarak tekrar yazılımı çalıştırıyoruz ve ayarlarımız düzgün mü öncelikle bunu kontrol ediyoruz. Hard disks paneline geçelim ve son olarak eklemiş olduğumuz System (DH0) disk bölümümüz bu panelde gözüküyor mu kontrol edelim. gözüküyor ise ayarlarda bir yanlışlık yok demektir.

Şekil 10.

Artık sistemi "start" tıklayarak başlatabiliriz. (Gözükmüyorsa konfigurasyon dosyasını yeniden gözden geçiriniz)



Şekil 11.

Sistemimiz doğrudan DH1 üzerindeki kurulu AmigaSYS sistemini boot edecek ve karşımıza Amiga'nın bilindik DOS ekranı çıkacak. Burada AmigaSYS'nin sistem ön yükleme ekranı bizi karşılıyor. Workbench 3.1 disketimizi yerleştirerek OS3.1 için gerekli dosyaları sistemimize kuracağız. E-UAE penceresindeki "Floppy Disks" paneline geçelim ve DF0: yazan birinci disket sürücümüze "insert" düğmesine tıklayarak gerekli bilgisayarımız bulunan daha önce temin ettiğimiz workbench disk imajlarından workbench31.adf disk imajını seçelim ve tamam diyelim.

Sistem yükleme ekranı karşımıza gelecektir, burada 1 yazıp enter'a basarak ingilizce AmigaSYS kurulumuna devam edebiliriz. (eğer elimizde Amiga Forever paketi varsa 4 seçerek Amiga Forever kurulumu da yapabiliriz) Kurulum ilerlediğinde bizden ekran çözünürlüğü seçmemizi isteyecektir. Burada da Picasso96: 800x600x32 ya da Picasso96: 1024x768x32 gibi bir çözünürlük seçerek devam ediniz. Keyboard layout sorulduğunda English seçerek devam ediniz. Sırada extras disketi var. Şimdi E-UAE'nin Floppy Disks panelinden extras.adf disket imajımızı yerleştiriyoruz ve Amiga ekranımızda enter'a basarak devam ediyoruz. Kurulum bitene kadar AmigaSYS ile ilgili çeşitli tanıtıcı ekranlar geliyor. Bu aşama da bittikten sonra extras disketimizi çıkarmamız ve Amiga'yı reboot etmemizi belirten bir mesaj ekrana gelecek. Yine Floppy panelinden disketimizi "eject" ediyoruz.

Amiga ekranına gelip enter'a bastığımızda sistemimiz boot reboot edecek ve işte karşımızda AmigaOS 3.1:)

Şekil 12.

Amiga OS 3.9 'a Yükseltme

Artık tek bir adım kaldı, Amiga OS 3.9 CD'imizi bilgisayarımıza takıp sistemimizi OS3.9'a yükseltmek. Bunun için Amiga sistemimizde öncelikle CD/DVD-Rom sürücümüzü mount etmemiz gerekmekte. Amiga ekranında iken Amiga + E tuşlarına basalım (Amiga'da Windows tuşunun yerinde A sembolü [yani Amiga

tuşu] bulunur. Dolayısıyla windows klavyelerde Windows + E tuşlarına basacağız) Ekranımıza bir komut satırı penceresi geldiğinde finddevice yazıp enter'a basalım. Amiga'nın device mount arabirimi gelecektir. Burada uaescsi.device seçeceğiz. Biraz beklediğimizde bize CD/DVD-Rom sürücümüzün hangi device üzerinde olduğunu belirtecektir. "Unit" sütununda belirtilen sayı bizim scsi.device olarak kullanımda olan cihaz numaramızdır ve bize gereken de aslında bu numaradır. Artık cihazımızın unit 0 olarak Amiga tarafından tanındığını bildiğimize göre mount ederek hemen kullanabiliriz.

Masaüstümüzdeki "System" sembolüne çift tıklıyoruz buradan "Devs" ve oradan da "DOS Drivers" ikonlarına çift tıklayarak açılan pencerenin içinde sağ tıklayıp "Window > Show > All Files" seçiyoruz. Burada normalde görünmeyen 2 klasör daha görünecek. CDR ve CDRoms. "CDRoms" klasörüne giriyoruz ve içerisinde bütün uniteler için mount scriptlerinin bulunduğu bir pencere geliyor. Benim sistemimde unit 0 cihazının sistemimdeki cdrom sürücü olduğunu bildiğim için "Unit0" simgesini çift tıklıyorum.

Komut satırı penceresi açılacak ve "Unit0" yazıyor olacak. Bu komut satırının en başına execute yazalım. Yani satır: execute unit0 olsun. Enter'a bastığımızda script çalıştırılacak ve böylece cdrom sürücümüz artık mount edilmiş durumda.

Şekil 13.

Çok kısa bir beklemenin ardından masaüstümüzde AmigaOS3.9 sembolünü göreceğiz ve otomatik olarak AmigaOS3.9 sürümüne sistemimizi yükseltmek isteyip istemediğimizi soran bir ekran gelecek.

Burada "Yes, now up to 3.9" tıklayalım.. Enter'a basarak kuruluma başlayabilirsiniz.

Şekil 14.

Karşınıza "Please insert volume OSFIR" yazan bir sistem requester penceresi gelecektir, burada "Deny" tıklayarak geçebilirsiniz. Cdrom sürücünüz gerekli sistem dosyalarını sabitdiskinize kopyalamaya başlayacaktır. Bu işlem birkaç dakika sürebilir. Sonunda karşınıza "Select AmigaSYS 3 Plus Theme" yazan bir requester açılacak. Buradan dilediğimiz bir sistem teması seçip kullanabiliriz. Ben sevdiğim temalardan "Gnome" seçtim, hangi ekran çözünürlüğünü tercih ettiğimizi soracaktır. Daha önce ekran çözünürlüğü olarak ne seçtiysek burada da onu seçelim ki temamız düzgün görünsün, ben 800x600 seçmiştim, o şekilde devam ediyoruz.

Şekil 15.

Ve işte kurulum bitti, enter'a basarak sanal Amigamızı reboot ediyoruz. Amiga sistemimiz tekrar açıldığında karşımızda OS3.9 masaüstünü karşımızda olacak.

Amiga OS3.9 çıktıktan sonra bazı güncellemeler yayınlanmıştı, bu güncellemeleri de indirip sisteminize kurmanızı tavsiye ederim.

http://os.amigaworld.de/index.php?lang=en&page=12 adresinden;

1. BoingBag 1
2. BoingBag 2
3. BoingBag 2 Contribution
4. BoingBag 2 Locale (turkce)

dosyalarını indirin, daha sonra bu dosyaları e-UAE'nin kurulu olduğu dizindeki AmigaSYS3Plus/Work/Archives/ dizinine kopyalayın. Böylece AmigaOS masaüstünüzdeki Work disk bölümünde bulunan Archives klasöründen bu güncelleme dosyalarına erişebileceksiniz.

Önce sisteminizde cdrom'u mount edin. Bunun için masaüstünüzde boş bir yerde sağ tıklayın, Tools menusunden CD-Rom Mount seçin ve device unit numarasını seçin "Mount" tıklayın. (benim sistemimde Unit0 idi hatırlarsanız) Böylece cdrom mount edildi, AmigaOS 3.9 CD'si sürücümüzde olsun, bundan sonra güncellemelere geçebiliriz.

Güncellemeleri sırasıyla yapınız, öncelikle BoingBag39-1.lha dosyasını çift tıklayın, karşınıza Packmaster programının penceresi gelince "Unpack" düğmesini tıklayın. Arşiv dosyası Ram Disk'e açılacaktır. Bundan sonra Ram Disk'e girerek BoingBag3.9-1 klasörüne girin ve install simgesini çalıştırın. Bütün sorulara "Proceed" diyebilirsiniz, dil seçim ekranı geldiğinde Türkçe'yi işaretleyin, devam edin. Güncelleme bittiğinde sistemi reboot edin. Tekrar açıldığında aynı işlemleri yaparak BoingBag39-2.lha dosyasını kurun, reboot edin. Bundan sonra BB3.9-2-Contribution.lha dosyasını kurabilirsiniz, bu opsiyonel bazı yazılımlar içermektedir, kurmanız şart değil. En son olarak da Türkçe dil güncellemesi için BB3.9-2-turkce.lha dosyasını kurunuz. Böylece sistemimiz gerekli güncellemeleri de içerir hale gelmiş oldu.

Artık mevcut işletim sisteminiz üzerinde çalışan sanal bir OS3.9 kurulu Amiga'nız var, güle güle kullanın:)

Önümüzdeki sayılarda bu konuda detaylı bilgiler vermeye devam edeceğim, yazı hakkında takıldığınız ya da merak ettiğiniz konuları bana e-posta ile yazabilirsiniz:

Gelecek sayılarda görüşmek üzere...

datura (at) inbox (nokta) com

Şekil 16. Amiga 1200

Ek bilgi;

Amiga modellerinde fabrika çıkışı olarak gelmekte olan donanımsal özellikler tablosu

	Amiga 1000	Amiga 500	Amiga 500+	Amiga 600	Amiga 1200	Amiga 4000
İşlemci	68000	68000	68000	68000	68020	68040
Kickstart	1.2	1.3	2.0x	2.0x	3.x	3.x
Chip Mem	256 kb	512 kb	1 mb	1 mb	2 mb	2 mb
Chipset	OCS	OCS	ECS	ECS	AGA	AGA

Test Platformu - Bölüm 4

Emir 'Skate' Akaydın

Test Platformu dökümanlarında dördüncü ve "son" bölüme geldik. İlk üç sayıda çapraz geliştirmeyle ilgili bilmeniz gereken birçok temel kavramı irdeledik. Bundan sonrası sizin Commodore 64 üzerindeki yeteneklerinizi geliştirmeniz ve yeni efektler kodladıkça PC gibi platformlardan nasıl faydalanabileceğinizi daha iyi öğrenmenizden ibaret. Ama bu işi tamamen size bırakmadan önce son bir örnek göreceğiz. Bu seferki örneğimiz 90lı yıllarda Commodore 64 demolarında sıkça rastladığımız bir demo efekti olan $d017 sprite stretch efekti. Bir de buna yan border açma, renk geçişi ve sinüs efektleri eklenince ortaya hoş bir örnek çıkıyor.

DİKKAT!!! Aşağıda anlatılan kavramların birçoğu ileri derecede VIC (Commodore 64'ün grafik çipi) bilgisi gerektirmektedir. Eğer bu konularda başlangıç seviyesindeyseniz, belirli bir yerden sonra döküman sizin için gittikçe zevksizleşecektir. Bu durumda test platformu bölüm 1'den 3'e kadar olan kısım size daha çok hitap ediyor demektir. VIC konusunda temel bilgilere sahip olmadan bu dökümanı okumanız olumsuz sonuçlar doğurabilir.

Doğrudan Sonuca Gitmek

Bu son bölümde lafı hiç dolandırmadan yapmak istediğimiz efekti, efekt için çözmemiz gereken problemleri ve bunların PC kullanarak çözeceğimiz kısımlarını inceleyeceğiz. Ancak bu defaki efektimizin özelliği PC'ye ihtiyaç duymaması. PC'yi bu defa bir nevi pseudocode yazma ve bu kodu test etme aracı olarak kullanacağız. Böylece Commodore 64 üzerinde efektimizi kodlamaya başlamadan önce bazı kod parçalarımızın düzgün çalıştığından emin olacağız.

Efektimizde kullanacağımız öğeler:

1. 8 sprite'ı sağ ve sol ekran sınırlarına denk gelecek şekilde basma
2. Yan border açma
3. FLD (Flexible Line Distance)
4. $d017 stretch (Spriteları Y ekseninde uzatma)
5. Uzama (stretch) tablosu üretme (Lineer İnterpolasyon)
6. Yumuşak renk değiştirme
7. Yumuşak sinüs hareketleri için bir sinüs tablosu

Bunlar efekti elde etmemiz için yapmamız gerekenler. Peki hangileri bize ne tür problemler çıkarıyor ve nasıl çözüyoruz tek tek inceleyelim.

8 sprite

Aslında bu çok basit bir işmiş gibi görünse de bilinmesi gereken bir püf noktası var.

Şekil 1.

Öncelikle sprite koordinatları ($d000-$d00f) ve X koordinatlarının 9.bitleri ($d010) hakkında bilgi sahibi olmanız gerekiyor. Bu bilgiler programcının el kitabı ya da diğer birçok commodore 64 dökümanından öğrenilebileceği için bu noktada üzerinde çok fazla durmuyorum. Asıl bilinmesi gereken şey x = 0 koordinatına basılan bir spriteın Commodore 64 ekranının en solundan (borderlar da dahil olmak üzere) 8 pixel yana denk geldiğidir. Yani borderlar açıkken ekranın en soluna bir sprite basmak istediğinizde x = 0 demek yeterli olmaz. Negatif değerlere gitmemiz lazım. Bunu yapmanın yolu ise $d010 adresindeki 9.bitden geçiyor. Bu biti ilgili sprite için set etmek, yani 1 yapmak lazım. Eğer 0 numaralı sprite'ı kullanırsak 0.bit = 1 olduğundan;

```
lda #$01
sta $d010
```

dememiz gerekiyor. Ancak işin ilginç tarafı mantıken $d000'ın (0.sprite X koordinatı) $ff değerini aldığında $00'ın bir solundaki pozisyon olması gerekirken $ff değeri verdiğimizde (9.bit'i de set iken) spriteın ekrandan kaybolduğunu görürüz. Çünkü 0'ın solu $ff değil $f7'den başlar. Kısacası;

- $d010 = $00; $d000 = $00 => 0.Sprite ekranin en solundan 8 pixel sagda
- $d010 = $01; $d000 = $f7 => 0.Sprite ekranın en solundan 7 pixel sağda
- $d010 = $01; $d000 = $f0 => 0.Sprite ekranın en soluna yapışık

$d010 set iken X ekseninde genişletilmemiş yani normal boyutlardaki bir sprite ($d01d = 0) $f0-$d9 değer aralığında, X yönünde genişletilmiş sprite (0.sprite için $d01d = 1) $f0-$c1 aralığında ekranın solunda görüntülenebilir. Elbette ki $f0 değeri kü-

çüldükçe sprite ekranın solundan dışarı çıkmaya başlayacaktır. Sprite ekranın solundan çıktıktan sonra $78'den büyük değerler için görüntülenemez. Ancak $77 ve daha küçük değerlerde bu defa sprite ekranın sağından içeriye girecektir. Dolayısıyla ekranın sağına yapışık X ekseninde genişletilmemiş spritein X koordinat değeri $60, genişletilmişin ise $48 olmalıdır.

Bu yukarıdaki son paragrafları anlamanızın tek yolu pratik yapmaktır. Ancak bu pratiği yapabilmek için border açmayı da öğrenmeniz gerekmektedir.

Not: Vice emulatorü borderları soldan 16 pixel, sağdan 6 pixel eksik gösterir. TV ekranları da hemen hemen Vice ile aynı sonucu verecektir. Bu örnek Vice ve TV ekranlarında da düzgün görünecek şekilde ayarlanmıştır.

Yan border açma

Aslında yan border açma yapılması gereken açısından incelenecek olursa oldukça basit bir şeydir. Tek yapmanız gereken tarama 39.karakterin bulunduğu bölgeye geldiğinde $d016 adresini değiştirerek borderı daraltmak ve yeniden genişletmektir. Bu sayede border kapanması gereken yerde kapanamayacaktır ve sağ borderda o raster satırı ve sol borderda bir alttaki raster satırı açılacaktır. İşin zor olan kısmı, yan borderları açmak istediğimiz bölge boyunca bu işlemi her satırda yapmamız gerekmesidir. Ayrıca zamanlamalar bu bölgede çok kritik olduğundan normal şartlarda titremeyen (stable) raster rutini kullanmamız gerekir. Ancak birazdan göreceğimiz örnekte bu rutin kullanılmadan yan borderların açılabileceğine tanık olacağız.

Yan border açmakla ilgili bir diğer temel problem ise 8 satırda bir DMA satırlarına denk gelindiği bölgede zamanlamanın yetersiz kalmasıdır. Ancak örneğimizde FLD tekniğiyle bu DMA satırlarından karakter ekranını kullanmamak koşuluyla kurtulabiliyoruz. Zaten amacımız şu an için yalnızca spriteları kullanmak.

FLD (Flexible Line Distance)

FLD 80lerden kalma çok eski bir teknik olmasının yanı sıra FLI, AFLI, Line Chrunching ve benzeri birçok grafik modu ve efektin atasıdır. Efekt $d011 adresi üzerine kuruludur. Normalde her sekiz raster satırında bir "bad line" adı verilen raster satırlarına gelinir ve burada ekstra 40 cycle harcanarak iç ekrandaki karakter alanları çizdirilir. Ancak $d011'in alt 3 biti ekranı Y ekseninde kaydırdıkça DMA satırları da ekranla birlikte kayarlar.

Şekil 2.

FLD tekniği de bu özellikten yararlanarak $d011'i her satırda kaydırıp tam DMA oluşacağı yerde yeniden ilk pozisyona getirerek hiç "bad line" oluşmamasını sağlayan tekniktir. Bu sayede ekran FLD'nin uygulandığı bölge kadarlık bir alanda aşağıya kayacaktır. Ancak FLD bölgesinde raster satırları zamanlamalar açısından homojen olacağı için işimiz çok daha fazla kolaylaşacaktır.

Burada dikkat edilmesi gereken nokta satırın hangi bölgesinde $d011'in değiştireleceğidir. Farklı bölgelerde değiştirmek farklı efektler ortaya çıkarabilir. FLD elde etmek için ekranın sol bölgelerinde $d011'i değiştirmek gerekir.

$d017 stretch

Commodore 64'de spriteların çözünürlükleri tek renkli 24x21 ve 3 renkli 12x24'dür. Bu çözünürlük değiştirilemez ancak spritelar X ve Y eksenlerinde iki kat genişletilebilirler. X ekseni için $d01d adresinde genişletilmek istenilen sprite'ın bitini set etmek ve Y ekseni için ise aynı işlemi $d017 adresinde yapmak gerekir. Eğer 0.Sprite'ı X ve Y'de genişletmek istersek;

```
lda #$01
sta $d01d ; X ekseninde genislet
sta $d017 ; Y ekseninde genislet
```

dememiz gerekir. X ekseni için farklı olarak yapabileceğimiz çok birşey yok, yani ya sprite orjinal boyutunda olacaktır ya da iki kat genişletilmiş halde bulunacaktır.

Y eksenine gelince işler değişiyor. Çünkü her raster satırında $d017 adresine farklı değerler verme imkanımız var. Ve yine her zamanki gibi bir VIC hatasından yararlanarak spriteları 2 katdan daha az ya da daha fazla genişletme imkanı elde ediyoruz. Ya da diğer bir deyişle spriteların her satırını birbirinden bağımsız olarak istediğimiz kadar genişletebiliyoruz. Bunun için yapmamız gereken öncelikle $d017'de ilgili sprite'ın bitini temizleyip hemen ardından o biti yeniden set etmektir. Örnek olarak tüm spriteları birden Y ekseninde genişletmek istiyorsak $d017'e öncelikle $00 değeri verip, ardından $ff değeri vermemiz gerekiyor. Bunun için;

```
lda #$00
sta $d017
lda #$ff
sta $d017
```

yazabiliriz. Ancak daha pratik ve genel bir tanım ile "eor" komutunu kullanarak;

```
lda #$00
sta $d017
eor #$ff
sta $d017
```

dememiz daha uygun olacaktır. Elbetteki her satırda $00 değeri vermemiz demek yalnızca spriteın en üst satırının uzaması demektir. Dolayısıyla istediğimiz uzunluğa ulaştığı yerde $00 yerine $ff değerini vererek bir sonraki sprite satırının görüntülenmesi sağlanabilir. Nedeni ise $ff'ine $ff ile eorlanmasının sonucunun $00 olmasıdır. Kısacası uzama bu noktada sona erecektir. Her satırın uzama miktarını belirlemek için en mantıklı yol bir tablo kullanmaktır. Yani;

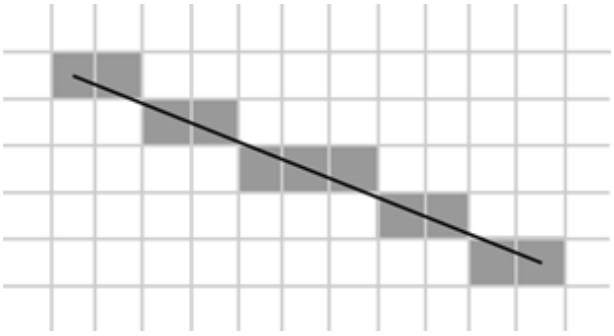
```
ldx #$00
loop:
...
lda stretch,x
sta $d017
eor #$ff
sta $d017
...
inx
cpx #63
bne loop

stretch
!byte $00,$00,$ff,$00,$00,$ff,$00,$00
!byte $ff,$00,$00,$ff,....,$00,$00,$ff
```

Yukarıdaki örnek sprite'ı tam 3 kat uzatacaktır.

Şimdi bir düşünelim. Bizim asıl elde etmek istediğimiz efekt ne? Spriteların orjinal yüksekliği 21 pixeldir. 21 pixelden başlayarak iç ekranın toplam yüksekliği olan 200 pixele kadar bu spriteları yumuşak bir şekilde uzatmak oldukça güzel görünecektir. Ancak bu iş için 21 ve 200'de dahil olmak üzere toplam 180 farklı tabloya ihtiyacımız olacak. Bu tabloları nasıl hesaplatacağız? Şimdi bu konuyu inceleyelim.

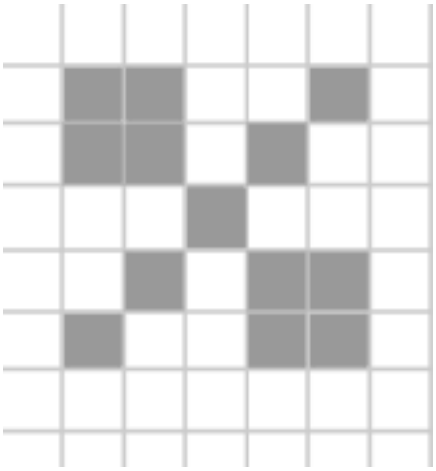
Uzama tablosu üretme

Eğer bu güne kadar çizgi (line) rutini kodlamışsanız bu problem size yabancı gelmeyecek demektir. Çizgi çekme rutininde ekrandaki A(x1,y1) ve B(x2,y2) noktaları arasında hangi adımlarla ilerlemeniz gerektiği sorusu temel problemi oluşturur. Bu konuda en çok kullanılan yöntemler Bresenham'ın çizgi algoritması ve bu algoritmanın üzerinden geliştirilmiş diğer algoritmalardır.

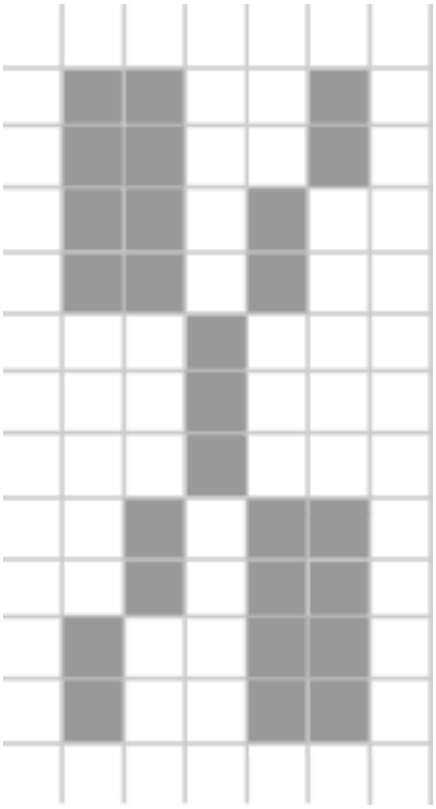


Şekil 3.

Uzatma rutini de temelde aynı mantıkla çözülebilir. Örneğin 5 pixel yüksekliğindeki bir resmi 11 pixele uzatmak istediğimizi varsayalım. Bu durumda yukarıdaki şekil bize her bir satırın kaçar kez tekrar etmesi gerektiğini verecektir. Yani;



Şekil 4. Orjinal resim



Şekil 5. Uzatılmış resim

Bu şekilde resmi uzatabiliriz. Şimdi Bresenham formülünü biraz daha basitleştirerek inceleyelim;

deltaO : Orjinal resmin yüksekliği (y2o-y1o farkı)

deltaU : Uzatılmış resmin yüksekliği (y2u-y1u farkı)

```
deltaO = y2o-y1o
deltaU = y2u-y1u
p = 0
for i from 1 to deltaU
    a = (deltaO / deltaU) * (i-1)
    if int(a) <> p then
        print "X"
        p = int(a)
    else
        print "o"
```

deltaO = 5 ve deltaU = 11 olduğuna göre adim değerini saydırarak genişletilmiş resmin hangi satırının orjinal resimin hangi satırına denk geleceği değerlerini elde edebiliriz. Ancak bunun için çarpma ve bölme işlemleri yapmak gerekiyor ki Commodore 64 için hiç de ideal bir yöntem değil bu. Çarpma işleminden kurtulalım;

```
deltaO = y2o-y1o
deltaU = y2u-y1u
adim = deltaO / deltaU
a = 0
for i from 1 to deltaU
    a = a + adim
    if int(a) <> p then
        print "X"
        p = int(a)
    else
        print "o"
```

Ancak hala bir problem var. "adim" değerini elde edebilmek için ondalıklı sayılarla çalışabilmemiz gerekiyor. Bizim yapmamız gereken tam sayılarla çalışabilecek bir formül bulmak. Neyse ki matematikçiler bunun da bir çözümünü bulmuşlar. Yine pseudocode yazacak olursak;

```
deltaO = y2o-y1o
deltaU = y2u-y1u
a = 0
for i from 1 to deltaU
    a = a + deltaO
    if a >= deltaU then
        a = a - deltaU : print "X"
    else
        print "o"
```

Dikkat ederseniz yazdığımız bu son pseudocode yalnızca toplama, çıkarma işlemleri ve koşullardan ibaret. Verdiği sonuç ise önceliklerle aynı. Bu durumda hadi bu formülün nasıl bir çıktı verdiğini test edelim. Bunun için istediğiniz dili kullanmakta serbestsiniz. Ben yine en pratik bulduğun Javascript dilini kullanacağım. Bu örnekte deltaO değerini spriteların yüksekliği olan 21 ve deltaU değerini örnek olarak 70 veriyorum. Yani 21 pixel yüksekliğindeki sprite 70 pixele uzatılsaydı her bir satır kaçar kez tekrar edecekti, bu script bize bunun sonucunu verecek.

lineer_interpolasyon1.htm

```
<script>
deltaO = 21;
deltaU = 70;
a = 0;
for(i = 0; i < deltaU; i++)
{
    a += deltaO;
    if(a >= deltaU)
    {
        a -= deltaU;
        document.write("X");
    }
    else
        document.write("o");
}
</script>
```

Bu programın çıktısı şu şekilde olacaktır.

```
oooXooXooXoooXooXooXoooXooXooXoooXoo...
```

Buradaki her X bir alt satıra geçilecek yeri ifade etmektedir. Yani bu çıktıya göre sırasıyla satırlar 4,3,3,4,3,3,4,3,3,4... kez tekrar edecektir. Başlangıçtan itibaren inceleyecek olursak 3 adet o ve ardından bir X gelmiş. Yani ilk satır 3+1=4 kez tekrar edecek. Ancak X'den sonraki satır artık orjinal resmin 2. satırı olacak anlamına geliyor.

Şimdi 21'den 200'e kadar elde etmek istediğimiz 180 tablo değerini hesaplayalım.

lineer_interpolasyon2.htm

```
<script>
str = "";
deltaO = 21;
for(deltaU = 21; deltaU <= 200; deltaU++)
{
    a = 0;
    for(i = 0; i < deltaU; i++)
    {
        a += deltaO;
        if(a >= deltaU)
        {
            a -= deltaU;
            str += "X";
        }
        else
            str += "o";
    }
    str += "<br>";
}
document.write(str);
</script>
```

Bu programın çıktısı bize istediğimiz tabloları verecektir. Bir önceki scripte göre hız optimizasyonu için her karakteri tek tek document.write ile ekrana çıktı verilmek yerine "str" ismindeki bir string değişkende biriktirip bir seferde tüm çıktıyı ekrana yazdırıyoruz. Bir diğer nokta ise her çıktının arasına
 koyarak çıktıların alt alta dizilmesini sağlıyoruz.

Aslında bu çıktıları "o" ve "X" yerine $00 ve $ff ile alsak ve kodumuzda doğrudan kullansak istediğimiz sonucu elde etmiş oluruz. Ancak buna gerek yok, bu kodun aynısını 6510 Assemblerdan da yazmak bizi hiç zorlamayacaktır. Burada çıktının doğru olduğundan emin olmuş olduk.

lineer_interpolasyon.a

```
        !to "lineer_interpolasyon.prg"

_A       = $f8
_DELTA_O = 21
_DELTA_U = 100
_TABLO   = $1000

        * = $0900
        ldy #$00
loop2   lda _A
        adc #_DELTA_O   ; a = a + deltaO
        sta _A
        cmp #_DELTA_U   ; if a < deltaU
        bcc pass1       ; then pass1
; a >= deltaU
        sbc #_DELTA_U
        sta _A          ; a = a - deltaU
        clc
        lda #$ff        ; tabloya $ff
        sta _TABLO,y    ; ekle
        iny
        cpy #_DELTA_U
        bne loop2
        beq out
; a < deltaU
pass1   lda #$00        ; tabloya $00
        sta _TABLO,y    ; ekle
        iny
        cpy #_DELTA_U
        bne loop2
out     rts
```

Bu programı yükleyip "SYS 2304" ile çalıştırdıktan sonra monitörden (Vice'da File->Monitor) "m 1000" yazarak 1000'den itibaren oluşturulan tabloyu görebilirsiniz.

Artık stretch efektinde gerekli tabloları gerçek zamanı olarak oluşturabiliriz.

Yumuşak renk değiştirme

Bir demoda seyirciyi en çok etkileyen özelliklerden biri de renk parlamaları (flashing colors), sıralı renk geçişleri (cycling colors), kayan renkler (scrolling colors) ve benzeri renk efektleridir. Spritelarımız uzama efektini yaparken bir taraftan da renk değiştirirlerse bu seyircinin gözüne hoş gözükecektir. Ancak sürekli renk değiştirmek yerine bir süre sabit bir renkte kalıp arada bir parlayarak başka bir renge dönüşmesi zaten yeterince hareketli olan efektimiz için iyi bir seçim olabilir. Bu durumda oluşturacağımız renk tablosu şu şekilde olmalıdır.

renk1,renk1,renk1,renk1,renk1,renk1,.....,geçiş renkleri,renk2,renk2,renk2,renk2,renk2,renk2,.....,geçiş renkleri

Şimdi ACME'nin !fill ve !byte komutlarını kullanarak 256 bytelık bir tablo oluşturalım.

```
flash   !fill 110,10
        !byte 12,12,12,15,15,15,15,13,13
        !byte 13,13,1,1,1,1,13,13,13
        !fill 110,14
        !byte 12,12,12,15,15,15,15,13
        !byte 13,13,13,1,1,1,1,15,15,15
```

Aralarda 18'erden toplam 36 geçiş rengi kullanıyoruz. Bu durumda sabit renkler de (256-36)/2'den 110'ar tane olarak hesaplanabilir. Böylece "!fill 110,10" ve "!fill 110,14" diyerek Açık Kırmızı (10) ve Açık Mavi (14) renklerinden 110'ar tane oluşturmuş oluyoruz. Aralarda ise geçiş renkleri var. Renkler hızlı değişeceği için ben renk geçişine pek özenmedim. Sizler isterseniz uygun renkleri peş peşe dizerek daha yumuşak geçişler elde edebilirsiniz.

Şimdi gelelim bu renkleri nasıl okutacağımıza. İstersek;

```
ldx flashCounter
lda flash,x
sta $d027
sta $d028
sta $d029
sta $d02a
sta $d02b
sta $d02c
sta $d02d
sta $d02e
```

diyerek tablodan okunan bir değeri tüm sprite renklerine atayabiliriz. Ancak bunun yerine;

```
ldx flashCounter
lda flash,x
sta $d02e
inx
lda flash,x
sta $d02d
inx
lda flash,x
sta $d02c
```

```
inx
lda flash,x
sta $d02b
inx
lda flash,x
sta $d02a
inx
lda flash,x
sta $d029
inx
lda flash,x
sta $d028
inx
lda flash,x
sta $d027
```

kodu bu şekilde kullanacak olursak spritelar tablodan peş peşe renkler okuyarak sıralı bir geçiş yaparlar. Bu sayede hem renk parlaması hem de renk kayması efektlerini bir arada kullanmış oluruz. Şimdi gelin bu uzun kodu ACME'nin özelliklerini kullanarak daha kısa ifade edelim.

```
ldx flashCounter
!for i,8 {
!if i != 1 {
inx
}
lda flash,x
sta $d027+8-i
}
```

Buradaki "!if" bloğunun amacı ilk seferde "inx" koymayıp, ondan sonraki tekrarlarda bu komutu aralara eklemek. Programın çıktısı birebir yukarıdaki açık örnekte olduğu gibi olacaktır.

Sinüs hareket tablosu

Uzama efektini eğer tek tek adımlarla ilerletecek olursak çok lineer bir görüntü elde ederiz. Uzamaya biraz hareket katmak maksadıyla güzel bir sinus tablosu oluşturmamız lazım. Ancak bu defa bu tabloyu da aynı renk tablosunda olduğu gibi arada biraz sabit bir değerde bekleyecek şekilde oluşturacağız. Bildiğiniz gibi normalde tablolarımızın uzunluğunu 256 bytelık seçeriz. Bu sayede bir bytelık bir sayacı sürekli arttırarak tablonun üzerinde kolaylıkla hareket ederiz ve başa dönmek için ekstra bir işlem yapmamıza gerek kalmaz. Yine 256 bytelık bir tablo kullanacağız. Ancak bu defa sinüs genişliğimiz 200 olacak. Geri kalan 56 byte ise sabit bir değerde olacak. Burada çözmemiz gereken problem sinüs tablosunun başladığı ve bittiği değerin bu 56 bytelık sabit değere eşit olması.

Bir diğer problem ise tek bir sinüs'ün efekte istenilen oranda hareket katmaması. Harmonik hareketi biraz daha hoş göstermek için sinüs * cosinüs gibi bir formül kullanabiliriz. Dikkat etmemiz gereken tek şey formüldeki tüm bileşenlerin peryodik sonuçlar vermesidir.

Sprite yüksekliği 21 ile 200 arası değişeceğine göre sinüs tablomuz da bu değer aralığında olmalı. Sabit bekleyeceği değeri 21 seçersek sinüs tablomuz 21'den başlayıp yine 21'de bitmelidir. Yani uzama efekti spriteın orjinal yüksekliğinden başlayacak ve bir süre bekledikten sonra gerçekleşecek. Sonra yine orjinal yüksekliğe geri dönecek.

Bu probleme örnek bir çözüm aşağıdadır.

sinus.htm

```
<script>
for(i = 0; i < 200; i++)
{
  a = 111
    + Math.floor(89.9
                 *Math.cos((i+50)/50*Math.PI)
                 *Math.sin((i+50)/100*Math.PI));
  if(i != 0)
    document.write(",");
  document.write(a);
}
</script>
```

Bu scriptin çıktısını programımıza şu şekilde eklememiz gerekiyor.

```
sinus   !fill 56,21
        !byte <sinus.htm'in ciktisini buraya kopyala>
```

Yukarıda Anlatılanların Bir Araya Gelmesiyle Oluşan "stretch.prg" Programının Çıktısı

Şekil 6.

Programın kaynak kodunu (stretch.a) ve çalıştırılabilir dosyasını (stretch.prg) disk magazinin paketinin içersinde bulabilirsiniz. Ayrıca "extra" klasörünün içersinde yukarıda örneklenen kodları da bulmanız mümkündür.

Kaynak kodun içersinde yukarıda anlatılanlara ek bazı bölümler de bulacaksınız. Hiç endişe etmeyin kodların tamamının açıklamaları yanlarından yer alıyor. Ancak elbette ki hiç Commodore 64 temeli olmayan biri için bunlar da yeterli olmayacaktır. Bu örneği tamamen anlayabilmek için özellikle VIC konusunda kendinizi geliştirmeniz şart. Bunun için Türkçe kaynak olarak size Nightlord'un yazdığı ve yakında sitesinden yayınlayacağı "C64 Grafik Programlama" serisini takip etmenizi tavsiye ederim. Link: http://www.nightnetwork.org/

Son Söz

Böylece "Test Platformu" serisinin sonuna gelmiş bulunuyoruz.

Aslında çapraz geliştirme kapsamlı bir konu ve anlatılabilecek çok fazla ileri düzey konu var. Örneğin Commodore 64 üzerinde bir vektör şehri yaratmak ya da basit bir video göstericisi programlamak. Ancak şu anki durumda bu son verdiğim örneğin bile Türkiye'de çok sınırlı bir kesime hitap edeceğini düşünüyorum. Dolayısıyla da daha ileri düzey konulara al atmak yerine, yeni bir yazı dizisine başlamak ve yalnızca çapraz geliştirmeye dayanan konularla sınırlı kalmamak bana daha uygun geliyor.

Özellikle bu son bölümü okuyan kişiler bu yazıyı biraz eksik bulabilirler. Evet, bazı kavramlar yeterince detaylı açıklanmıyor ancak bu yazının eksikliği değil, aksine fazlalığından kaynaklanıyor. Bu yazıda çapraz geliştirmeyle doğrudan ilgili kısımlar lineer interpolasyon hesapları ve sinüs hesapları. Bunun haricindeki kısımlar fazladan ve biraz da mecburiyetten anlatılmış bölümler. Aslında yan border açmayla ilgili kısıma da bir cycle hesaplayıcı eklemeyi düşündüm ancak bunun da şu an için kafa karıştırmaktan daha öteye gitmeyeceği kanısına vardım.

Özetlemek gerekirse "Yan Border Açma", "FLD", "$d017 Stretch", "Yumuşak Renk Değiştirme" konuları fazladan açıklanmış konulardır. Amaç, diğer dökümanlardan okuyup öğreneceğinizi varsaydığım bu konuları buradaki örnekle bağdaştırabilmenizi kolaylaştırmaktır.

Bu bölümün yazının son bölümü olması, Test Platformu - Bölüm 5'in olmayacağı anlamına gelse de, zamanla bana gelecek talepler doğrultusunda böyle bir devam bölümünün olma ihtimali de var. Yalnızca Plazma'da her sayı devam eden seri burada sona eriyor. Belki ilerki sayılarda yeni bir Test Platformu ile yeniden karşınıza çıkabilirim. Şimdi eğer hala denemediyseniz "stretch.prg" dosyasını emülatörden çalıştırın ve mini şovu izleyin :)

emir (at) akaydin (nokta) com

Vice ile Debug

Bilgem 'Nightlord' Çakır

Giriş

Merhaba arkadaşlar. Bu yazıda genel bazı hata arama ve bulma tekniklerinden bahsedip, bu tekniklerin özellikle C64 programları geliştirirken nasıl uygulandıklarını göstereceğiz. Bunun için en önemli araçlardan biri olan Vice emülatörünü kullanacağız. Vice emülatörü hem Windows hem Linux'ta çalışan bir araç olduğu için iki platformda Vice'ın farklı olduğu noktaları da belirteceğiz.

Bu yazıda aşağıdaki konular anlatılacak:

- Debuggerlara Genel Bakış
  - Breakpointler
  - Kod'u adım adım çalıştırma
  - Registerları inceleme
  - Belleği inceleme
- Vice Monitörü
  - Örnek kod
  - Monitöre giriş ve çıkış
  - Breakpointler
  - Adım adım ilerlerken registerları takip etmek
- C64 Programlarında Sık yapılan hatalar
- Daha ileri teknikler
  - Bellekteki değişmeleri yakalamak.
  - Karmaşık breakpointler
  - Başkalarının programlarını incelemek.

Debuggerlara Genel Bakış

Her platformda debugger deyince aklımıza gelen bazı ortak noktalar vardır. Genelde yazılan bir programın içinde herhangi bir anda belleğin ve CPU registerlerinin durumunu sorgulayarak o programın içindeki hataları bulmaya ve gidermeye çalışırız. Debuggerlar çeşitli GUI'ler veya komut konsolu arayüzlerine sahip olabilir. Fakat bazı temel özellikler bütün debuggerlarda aynıdır.

- Breakpointler

Bellekte çalışan kodun, herhangi bir noktasına gelince programın duraklamasını isteyebiliriz. Breakpointler debuggerda, bellekteki kodun bir satırına (veya makine dili komutuna) yerleştirebildiğimiz bir araçtır. CPU kodu işletirken, breakpoint ile karşılaşırsa o anda programı dondurup kontrolu bize ve debugger'a verir. Böylece biz debugger'ın çeşitli komutlarını kullanarak registerlerın ve belleğin tam o anda sahip oldukları değerleri inceleyebiliriz.

Genelde breakpointleri kullanırken aklımızda iki şey vardır:

Birincisi CPU oradan önceki komutları başarıyla işletip bu noktaya gelebiliyor mu, bunu görmek isteyebiliriz. Programımızı yazıp çalıştırdığımız zaman ilk karşımıza çıkan ve yanlış olan belirti, genelde bize nereye bakmak istediğimiz ile ilgili fikir verebilir. Örneğin, programın bir yerde ekrana bir mesaj çıkarmasını bekliyorsak ve bu mesaj çıkmıyorsa, mesajı yazdıran fonksiyonu çağıran yerlere breakpoint koyabiliriz. Programımız o fonksiyonu düzgünce çağırıyor mu? Bazen bu şekilde breakpoint koyup programımızı çalıştırınca CPUnun durmadığını görür ve anlarız ki program oraya hiç gelmiyor. Ya bir yerlerdeki kontrollü dallanma (if else / je / bne ...) bizim beklediğimiz gibi çalışmıyor, ya da program daha önce bir yerde bir sonsuz döngüde kalıyor.

İkincisi ise eğer program başarıyla breakpointe ulaşıp duruyorsa breakpointten sonraki o kritik algoritmamızı yavaş yavaş inceleyebilme isteği. Çoğu zaman bir breakpointten sonra programımızı adım adım çalıştırarak yaptığımız pekçok hatayı buluruz.

Breakpointleri öğrendiğiniz anda hata ayıklama sanatının %50'sini öğrendiniz demektir.

- Kod'u adım adım çalıştırma

Genelde bir algoritmanın tam olarak doğru çalışıp çalışmadığını anlamak için debugger'ın adımlama özelliklerinden faydalanırız. Bütün debuggerlarda Step-In, Step-Over ve Return-from-Function denilen üç adım çeşidi her zaman vardır.

Step-In bir seferde bir komut ilerlemenizi sağlar. Bu komut eğer bir alt fonksiyona atlama komutu ise, Step-In yaptığınızda gidilen alt fonksiyonun ilk komutunda bulursunuz kendinizi.

Step-Over da bir seferde bir komut ilerlemenizi sağlar. Fakat Step-In'den farklı olarak eğer bir sonraki komut alt rutine atlama komutuysa, Step-Over ile bütün alt fonksiyonu bir seferde çalıştırıp o alt rutin dondukten sonraki ilk komutta bulursunuz kendinizi. Dolayısıyla kodu adım adım ilerletirken detaylı incelemek istemediğiniz bir alt rutin çağrısı ile karşılaşırsanız Step-Over ile vakit kaybetmeden ilerleyebilirsiniz.

Son olarak Return-From-Function komutu ile de o an bulunduğunuz fonksiyon veya rutinin sonuna kadar ilerleyip, o rutini çağıran yere dönebilirsiniz. Bu da genelde bir alt rutini incelemek için Step-In ile girdikten sonra aradığınız sonucun o rutinde olmadığını görünce kullanabileceğiniz bir komuttur.

- Registerları ve Belleği inceleme

Genelde kodda adım adım ilerlerken her komuttan sonra CPU registerlerindeki değerleri inceleyerek komutların ve hesaplamaların beklediğiniz sonuçları ortaya çıkarıp çıkarmadığını takip edebilirsiniz. Her debugger CPU registerlerindeki değerleri ve belleğin durumunu size gösterecek komutlarla donatılmıştır.

Vice Monitörü

Bu kadar genel tanıtımdan sonra yukarıda bahsettiğimiz komutları Vice'da nasıl vereceğimizi inceleyelim. Bu bölümde örnek olarak hatalı bir kod ile başlayacağız. Daha sonra vice debugger'ını kullanarak hatayı nasıl bulabileceğimizi göreceğiz.

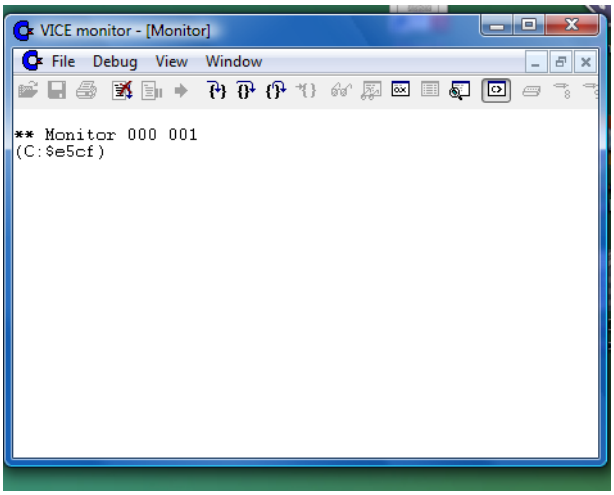
- Örnek kod

Bir bahar akşamı cevval coder adayı Psychadelic/Granite yeni introsunu (Devil is My Friend adlı şaheser olacaktı) kodlamaya başladı. Tabii ki ilk olarak her introsunda olduğu gibi ekrana bir tane raster çizgisi koyarak başlayacaktı. Hemen aşağıdaki kodu yazdı. Birkaç ACME uyarısını atlatıp sonunda kodu derledi. Vice ile açtı. sys 49152... Ve hic bir şey olmadı.

```
;; UYARI: BU KOD B#L#NÇL# OLARAK
;; HATALAR #ÇERMEKTED#R

        !to "devil.prg", cbm
        *=$c000
;; -----------------------------------
Baslangic:
        jsr RasterIRQHazirla
Son:
        jmp Son
;; -----------------------------------
RasterIRQHazirla:
        sei

        lda #$7f
        sta $dc0d

        lda $d01a
        ora #$01
        sta $d01a

        lda $d011
        and #$7f
        sta $d011

        lda #$20
        sta $d012

        lda $00
        sta $0314
        lda $c1
        sta $0315

        cli
        rts
;; -----------------------------------
        *=$c100
IRQRutini:
        inc $d019

        ldx #$2
IRQ_gecikme_dongusu:
        dex
        bne IRQ_gecikme_dongusu

        lda $d020
        pha
        lda #$01
        sta $d020

        ldx $0a
IRQ_gecikme_dongusu2:
        dex
        bne IRQ_gecikme_dongusu2

        lda #$0e
        sta $d020

        jmp $ea81
;; -----------------------------------
```

- Monitöre giriş ve çıkış

Bunun üzerine Psychadelic hemen Vice'ı resetleyip bu sefer sys49152 demeden önce, Alt-M ile Monitore girdi (Linux Alt-H). Artık Debugger konsolu karşısında onun komutlarını bekliyordu.



Şekil 1. Vice Debugger Konsolu

- Breakpointler

Hemen kendine sordu. "Kurduğum interrupt çalışıyor mu? Program interrupt rutinine ulaşıyor mu?" Bunu anlamak için $c100'deki interrupt rutininin başına bir breakpoint koymaya karar verdi. Bunun için konsola şu komutu yazdı

```
break c100
```

Ardından kodu çalıstırmak için debuggerdan çıkmaya gerek duymadı ve direk başlangıç adresine atlama komutu verdi.

```
g c000
```

Program breakpointte durmadı ve çalışmadı. Psychadelic anladı ki interrupt rutinini doğru kurmuyor. Bunun üzerine programın başına breakpoint koyup interrupt rutinini hazırladığı bölümü tek tek incelemeye karar verdi.

```
break c000
g c000
```

İkinci komutu vermesiyle beraber program calışmaya başlamak uzere c000a atladı ve oradaki breakpointe yakalandı

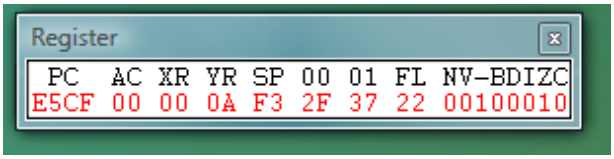
- Adım adım ilerlerken registerları

takip etmek

z komutunu kullanarak adım adım komutları işletmeye başladı. Vice kullanıcı konsol penceresinde Enter'a her bastığında son debugger komutunu işletiyordu. Bu sayede ilk seferde z komutunu verdikten sonra her enter'a basışında yeni bir Step-In komutu vermiş oluyordu. Bu komutlar sayesinde program satır satır ilerledi ve her defasında bir sonraki assembly komutunun işletildiğini gördü.

Sonsuz döngü komutuna kadar böyle gelince, buradaki komutların hepsine uğrandığına dair emin olduktan sonra daha detaylı incelemeye karar verdi. tekrar g c000 yaparak başa döndü.

Bu sefer debugger'ın registers penceresini de açtı. Bütün komutları tek tek işletirken bir yandan da her komuttan sonra registerlerın değerlerine bakıyordu. (Vice'ın linux versiyonunda malesef ayrı bir register penceresi yoktur. Debugger konsolunda registers komutu vererek registerlar listelenebilir. Dolayısıyla satır satır ilerleyip her satırdan sonra registerlara bakma için sırayla bir Step-In bir registers komutu vermek gerekiyor. Neyseki ilk defa bu komutları kullandıktan sonra, yukarı ok tuşunu kullanarak son kullanılan komutlara erişmek mümkün)



Şekil 2. Debugger Registers Penceresi

Sonunda tam sta $0314 komutuna geldiğinde A registerinde garip bir değer gördü. Daha bir önceki satırda 0 yüklenmesine rağmen şimdi A registerinde $2f değeri vardı. bir önceki satıra tekrar baktı:

```
lda $00
```

Birden kafasında şimşek çaktı. Bu komut akümülatöre 0 yüklemiyordu. Aksine 0. adresten okuduğu değeri yüklüyordu. Yani adresleme modu yanlıştı. Immediate adresleme modu yerine sıfırıncı sayfa adresleme modu çalışıyordu. Hemen komutu düzeltti.

```
lda #$00
```

Zaten bir altındaki satırda da aynı hatayı yaptığını farketti ve onuda düzeltti. Hemen acme ile yeniden derleyip vice ile çalıştırdı.

Sonuç hüsrandı. Yine boş ekran ve ready yazısına dönmüştü makine. Hemen çıkıp bir daha derledi ve çalıştırdı sonuç aynıydı. Yalnız bu sefer ekranda ready mesajından önce küçük beyaz bir flaş olduğunu farketti. Bu iyiye işaretti. Acaba bir şekilde IRQ rutinindeki beyaz çizgi koduna varıyor muydu sistem.

Hemen vice monitörüne girip yine IRQ rutinine breakpoint koydu ve programı çalıştırdı.

```
break c100
g c000
```

Debugger breakpointte durdu. Yani sistem başarıyla interrupta gelebilmişti. Sistem interruptı düzgün kurduktan sonra her ekran taramasında yeniden c100'deki breakpointe yakalanmalıydı. Daha detaylı bakmadan önce tekrar yakalanacak mı diye kontrol etmek için programı devam ettirmeye karar verdi. Bunun için debuggerdan çıkış komutunu verdi:

```
exit
```

bu komutla beraber program devam etti ve kendisini ready yazısında buldu. Demek ki IRQ rutinine sadece bir kere giriliyor sonra IRQ rutininde yaşanan bir problemden dolayı bir daha girilemiyordu. Bunun üzerine IRQ'dan çıkışı kontrol etmeye karar verdi. Alt-M ile tekrar monitöre girip kernelde atladığı IRQ çıkış rutinine breakpoint koydu.

```
break $ea81
g c000
```

Program önce daha önceden koyduğu c100'deki breakpointte durdu.

```
exit
```

Bu sefer program ea81 adresinde durdu. Burada art arda z komutuyla ilerlerken karşısına RTI komutu çıktı. Bu komutla CPU IRQ işlemekten çıkacak ve normal rutine yani bu programda ihtimalen c003 adresindeki sonsuz döngüye dönecekti.

Fakat z demesiyle beraber kendini alakasız bir adreste buldu. Karşısına BRK komutu çıkmıştı. Yani program IRQ dan düzgün şekilde dönmüyordu.

İnterrupttan dönerken olanları düşündü. RTI komutuyla işlemci stackten döneceği adresi alıyor ve oraya atlıyordu. Bu program yanlış yere atladığına göre stackten yanlış adresi alıyor olmalıydı. Ama stack nasıl bozulabilirdi. Hemen IRQ rutini için yazdığı koda bir daha baktı. Ve orada PHA komutunu gördü.

"Aaaah ben d020'deki değeri stack'e atıp beyaz çizgiden sonra geri stackten okuyacaktım." Evet başta böyle tasarlamıştı. Ama sonra bunu unutup d020 yi eski rengine döndürmek için PLA ile stackten değeri okuyacağına LDA ile direk yüklemeye kalkmıştı. Bunun sonucu olarak, IRQya girilirken stackteki en üst iki bayt dönüş adresi oluyor fakat çıkış esnasında aradaki PHA ile gonderilen bayt hiç PLA ile geri çekilmediği için en üst iki bayt dönüş adresi baytlarından biri ile d020'nin PHA'lanmış rengi oluyordu. Bu da yanlış adres demekti. Hemen gidip lda #$0e satırını sildi ve yerine pla komutunu koydu:

```
pla
sta $d020
```

Hemen derleyip yeniden çalıştırdı. Ve bu sefer karşısına beyaz bir çizgi çıktı. İşte artık IRQ çalışıyordu. Fakat çizgi çok kalındı.

Gecikme değeri olarak sadece $0a koymuştu oysaki. Ama artık içi rahattı. Bu raster çizgisinin çıkmamasına göre çok daha küçük bir hataydı.Elbet bulunurdu.

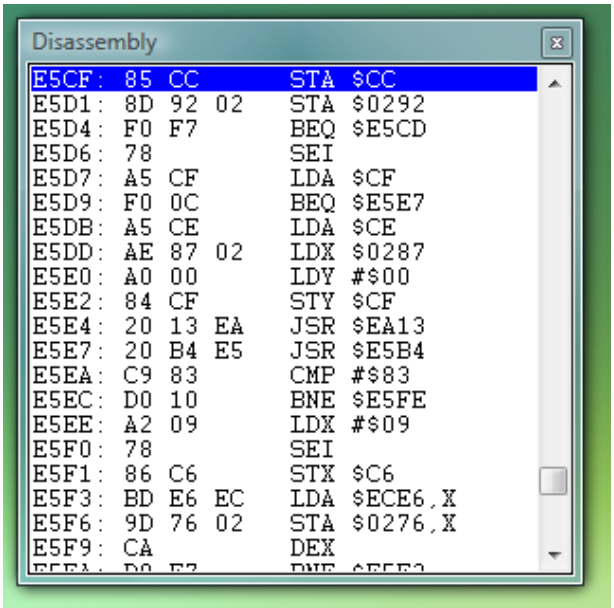
Hemen gecikme döngüsünü tekrar inceledi. çok geçmeden hatayı gördü. Yine # karakterini unutmuş ve yanlış adresleme modunu kullanmıştı.

```
lda #$0a
```

ve tekrar derlediğinde artık programı çalışıyordu.

Vice komutlarının özeti

- break c000 : $c000 adresine breakpoint koy
- g c000 : $c000adresinden kodu çalıştırmaya başla
- z : (breakpointe gelip durduğunda) Step-In
- n : Step-Over
- ret : Fonksiyondan geri dön
- exit : debuggerdan çıkıp sonraki breakpointe rastlayana kadar normal çalışmaya devam et
- registers: registerların o anki değerini göster
- m c000 : c000 adresinden başlayarak bellekteki baytları göster
- d c000 : c000 adresinden başlayarak assembly kodunu listele



Şekil 3. Disassembly Penceresi

Sık Yapılan Hatalar

Bazı basit hataları nedense en tecrübeli programcılar bile sıklıkla tekrar tekrar yaparlar. Bunlardan bazıları aşağıda. Eğer programınız çalışmıyorsa ilk arayacağınız hatalar bunlar olmalı. Yaptığınız hataların en az %50'si bunlar olacaktır.

- Unutulan #: Bunu örneğimizde de gördük
- Stack hataları: Dengesiz push ve pop komutları. Eğer bir IRQ rutininde veya alt rutinde pha ve pla komutlarının sayısı eşit değil ise, yani başka bir deyişle stack'e PHA ile koyduğunuz her değeri geri almıyor veya fazladan değerler alıyorsanız. Programınız RTS veya RTI komutuna rastladığında yanlış yere atlayacaktır. Bunun sebebi de daha önce de belirttiğimiz gibi dönüş adresinin stackte tutulması.
- İndex hataları: İndexli adresleme kullanırken (lda $c800,x gibi) iki şeyi kontrol ettiğinizden emin olun. Bu komutlar taban adresinin yanlış olduğu durumlarda veya X registerinin beklenmedik bir değer aldığı durumlarda yanlış çalışırlar. Genelde okunan tablo ebadı gözden kaçabilir. Örneğin c800 adresinden başlayan 32 baytlık bir tablonuz varsa X registeri de 0 ile 31 arasında olmalıdır. Eğer X registerindeki değer başka bir matematiksel işlemle hesaplanıyorsa bu işlemin sonucunun 0-31 arasında kalıp kalmadığını kontrol etmelisiniz
- Karşılaştırma hataları: Özellikle C64 ile ilk programlamaya başladığınız zamanlarda koşullu dallanma komutlarını birbiriyle karıştırabilirsiniz. BNE, BEQ en çok kullandıklarınız olduğu için fazla karışmaz fakat, BPL ile BCCyi karıştırabilirsiniz bu komutların anlamlarını tam olarak anladığınızdan emin olun
- Unutulan inc $d019: VIC IRQlarını kullanırken, IRQ rutini içinde inc d019 ile VIC'e IRQ isteminin işlendiğini bildirmezseniz IRQ rutininiz sadece bir kere çalışır
- Kendini değiştiren programlar yazarken yanlış baytı değiştirme: Genelde C64de hız optimizasyonu olarak kendini değiştiren kodlar yazmak sık kullanılan bir tekniktir. Bunların debug edilmesi genelde daha zordur. Eğer programınız beklemediğiniz şekillerde kilitleniyorsa kodun kendi kendini değiştirdiği noktaların yakınlarına breakpointler koyarak oraları adımlamalısınız. Örneğin aşağıdaki koda bakın

```
ldx value
lda jmpTable_lo, x
sta jumper
lda jmpTable_hi, x
sta jumper + 1

jumper:
jmp $0000
```

Bu kod val değerine bakarak bir adrese atlamaya karar veriyor. Yani değişik val değerlerine göre değişik yerlere atlayacak. Fakat bunu denerseniz çalışmayacak ve debug ederseniz, jmp komutunun bozulduğunu göreceksiniz. Bunun sebebi jumper adresindeki ilk baytın JMP komutunun kendisi olması. Değişmesi gereken baytlar aslında jumper ve

jumper+1 adreslerinde değil, jumper+1 ve jumper+2 adreslerinde yer alıyor.

- 16 bitlik değerlerin üst 8 biti ile ilgili hatalar: Bazen 16 bitlik değerlerle uğraşırken hatalar yapabilirsiniz. İlk önce her zaman iki baytı aynı şekilde anlamlandırdığınızdan emin olun (küçük bayt önce büyük bayt sonra yani little endian). Bunun dışında 16 bitlik aritmetik işlemlerde carry bitlerini doğru kullandığınızdan emin olun

Daha ileri teknikler

Önceden de belirttiğimiz gibi en çok yapılan hatalar ve temel breakpoint koyma ve kodu adımlama ile ilgili bilgileriniz zaten debug işlemlerinizin %80'ini oluşturacaktır. Bunun yanında elbette bazı daha ileri teknikler de mevcut. Bunların kullanımı yer yer daha karmaşık olabilir fakat zaman zaman ihtiyaç duyabilirsiniz

- Bellek erişimlerini yakalamak.

Bunun için breakpointlere benzeyen başka bir aracı kullanıyoruz. Vice'da bunlara "watchpoint" deniyor. Bellekte bir adrese veya adres aralığına watchpoint koyarak, oranın okunduğu veya yazıldığı anda programın bir breakpointe gelmiş gibi durmasını sağlayabiliriz. Bu özellikle bellekte bir bölgeye koyduğunuz kod veya data siz istemeden üzerine yazılıyorsa ve siz bunu yapanın kim olduğunu bulmak istiyorsanız çok işinize yarayacaktır.

```
watch store $0314
```

Bu komut 0314'e bir değer yazılınca durulmasını sağlar.

```
watch load $1000
```

bu komut 1000 adresinden okuma yapınca durulmasını sağlar.

- Trace

Bu da breakpoint ve watchpointlere benzer şekilde "tracepoint" adı verilen bir aracın kullanılmasıyla olur. Yine istediğiniz adreslere tracepointler koyarsınız. Ardından programı çalıştırdığınızda CPU tracepointlere rastladıkça adresini yazar. Böylece bu yazılan adreslere bakarak programın çalışırken nerelerden geçtiğini görmüş olursunuz. Yani programın yürüdüğü yolu takip edebilirsiniz. Bu da genelde kilitlenmelerde işinize yarar. Kilitlenmeden önce programınızın hangi noktalardan geçtiğini öğrenebilirsiniz. Bunun için trace komutunu kullanmanız yeterlidir

```
trace c000
```

Bu komut c000 adresine bir tracepoint koyar

- Karmaşık breakpointler

Bir diğer konu breakpointlerinizi koşullu yapabilirsiniz. Bazen büyük döngülerin içinde her defasında uğramak istemediğiniz yerlere breakpoint koymanız gerekir. Bu zamanlarda breakpoint'in sadece belli koşullarda durmasını sağlayabilirsiniz. Bunun için condition komutu kullanılır. Önce normal şekilde breakpoint konulur. Bu noktada oluşturulan breakpointin numarasına bakılır. Ardından aşağıdaki şekilde koşul tanımlanır:

```
condition 2 if .X==7
```

Bu örnek 2 numaralı breakpointin yalnızca X registerinde 7 değeri varken programı durdurmasını sağlar

Başkalarının programlarını incelemek.

Son olarak, çok faydalı bir öğrenme yolu olan başkalarının kodunu dissassemble etmekten kısaca bahsedeceğiz. Genelde C64 te hoşunuza giden ilginç bir demo efekti veya eklentiler yapmak istediğiniz toollar olabilir. Bunları disassemblerda inceleyip anlamak sabır ve tecrübe ister ama bir yerlerden başlamak gerekir. İşte başlangıç için bir kaç ipucu

- h (Hunt) komutu: Bu komut ile bir adres aralığında istediğiniz uzunlukta bir bayt dizisini arayabilirsiniz. Örneğin

  ```
  h 0800 d000 78 a9
  ```

  komutuyla 0800 d000 adresleri arasında 78 a9 baytlarının arka arkaya geldiği yerleri ararsınız. Neden 78 a9 örneği verdik. Çünkü 78 sei komutudur a9 ise lda# komutudur. Bu komutla bellekte bir demo rutininin IRQ hazırlığını yaptığı bölümü bulabilirsiniz. Ardından çıkan adresteki kodu disassemble ederek takibe başlayabilirsiniz. Böyle başka aramak isteyebileceğiniz komutlar sta $0314, sta $d012 olabilir. Aynı şekilde ilginizi çeken bir rutin bulduktan sonra o rutinin adresine nereden jsr veya jmp yapıldığını da arayabilirsiniz.

- m d000 komutu: Bu komut daha önce anlattığımız m komutu yani bellek adreslerine bakmak için. Tabi çoğu zaman d000 adresinde IO bölgesi aktif olduğu ve VIC çipinin registerleri orada olduğu için m d000 komutu çok faydalıdır. Bize VIC çipinin o anki durumunu verir. Mesela d015 adresine bakarak spriteların kulllanılıp kullanılmadığını, d018 adresine bakarak video matris ve karakter bankları için hangi adreslerin kullanıldığını öğrenebilirsiniz.

- sta d011 aramak: Demin bahsettiğimiz h komutuyla d011 erişimlerini arayabilirsiniz. Neredeyse bütün VIC efektlerinde d011 erişimleri kullanıldığı için bu şekilde kolaylıkla IRQ rutinini bulabilir ve Crossbow'un cycle artırma tekniklerini öğrenebilirsiniz.

- defter ve "çağırma grafiği": Son olarak eğer bir programı tamamen anlamak istiyorsanız, detaylı bir şekilde hangi adresteki hangi rutin kimi çağırıyor oturup bir deftere çizelge olarak çizmeniz gerekir. Malesef genel olarak VIC efektleri dışında komplike araçları oyunları veya matematiksel demo efektle-

rini bu şekilde anlamak hayli zordur. Ama tecrübeniz arttıkça şansınız artacaktır.

Sonuç

Bu yazıda oldukça geniş bir konu olan hata bulma tekniklerine değinmeye çalıştık. Artık C64 üzerinde kod geliştirirken daha donanımlı olacağınızı ve elinizdeki araçlardan daha fazla verim alacağınızı umut ediyoruz.

Bugsız günler

nightlord (at) nightnetwork (nokta) org

Yararlanilan Kaynaklar

- Vice Online Documentation: http://www.viceteam.org/vice_toc.html

Amiga Assembly Kursu

Şemseddin 'Endo' Moldibi

Şekil 1.

Merhaba, bu sayıdan itibaren sizlerle birlikte Amiga Assembly ile nasıl program yazacağımızı öğreneceğiz. Amiga Assembly ile program yazmak son derece eğlenceli ve kolaydır. Basit bir Assembly kodu aşağıdaki gibidir:

Örnek bir program

```
;basit bir kontrol toplami (checksum) hesaplama
start:
  move.l  #0,d0         ;D0 register'ine
                        ;0 koy

  move.l  #0,d1         ;D1 register'ine
                        ;0 koy

  lea   data(pc),a0 ;data'nin adresini
                        ;A0'a al. A0 data'ya
                        ;bakan bir pointer
                        ;olmus oluyor

.loop:
  move.b  (a0)+,d1      ;A0'in gosterdigi
                        ;adresten bayt olarak
                        ;oku ve D1'e koy
                        ;A0'i (pointer) bir
                        ;arttir

        add.l d1,d0 ;D1'i longword olarak
                        ;D0'a ekle

        cmp.b #0,d1 ;D1 ile sifiri
                        ;karsilastir

        bne.s .loop ;D1 sifir degilse
                        ;.loop'a git

        move.l d0,checksum
                        ;D0'in degerini
                        ;checksum'a yaz

        rts       ;programdan cik

data:
        dc.b 'plazma',0
                        ;bayt olarak plazma
                        ;yazisi

        even            ;bir sonraki adresi
                        ;cift yap

checksum:
        dc.l 0      ;checksum'i tutacak
                        ;bir longword
```

Peki bu kodu nasıl derleyeceğiz? Bunun için Amiga'da kullanabileceğiniz pek çok assembler vardır, ben yazılarımda AsmPro V1.17 kullanacağım. Sizler dilerseniz AsmOne, PhxAss ya da başka bir derleyici kullanabilirsiniz. İleriki sayılarda diğer derleyicilere de değineceğiz.

Yukarıdaki kodu derlemek için sisteminizde AsmPro olmalı, eğer yoksa http://www.aminet.net adresinden AsmPro kelimesini aratarak bulabilirsiniz. Aminet'ten indirdiğiniz lha arşivini bir yere açın ve AsmPro'yu çalıştırın.

Sizden "AsmPro:" isimli disketi yerleştirmenizi isteyebilir, bu noktada CLI'den (shell penceresi) "Assign AsmPro: <AsmPro'nun-bulunduğu-klasör>" ile AsmPro'nun olduğu yeri gösterebilirsiniz. Dilerseniz "cancel" deyip geçebilirsiniz. Bu ve benzeri AmigaOS konularını Plazma'nın ileriki sayılarında okuyabilirsiniz.

Nerede kalmıştık? Kodu derlemek için AsmPro'yu açtınız, karşınıza workspace'iniz (çalışma alanınız) için kullanacağınız RAM tipi ve miktarını gireceğiniz bir ekran gelecektir. Burada Fastmem veya Chipmem'i işaretleyip miktar için ~100 KB seçebilirsiniz. (Daha fazla da olabilir) Burada bir hatırlatma yapalım seçtiğimiz miktar sadece derlenmiş program için değil tüm çalışmamız için kullanılacaktır, yani üzerinde çalıştığımız kaynak kodlar (source), derleme işleminde kullanılan etiketler (label) vb., o yüzden biraz yüksek tutmakta fayda vardır.

Devam edelim, herhangi bir şey yazmıyorken Enter tuşuna bastığınızda AsmPro ekranı temizler. Escape tuşu ile hızlıca Editör'e girip çıkabilirsiniz. Bir kez Escape tuşuna basarak editöre girin ve yukarıdaki kodu aynen yazın. Burada kod satırlarının sağ tarafındaki açıklama (comment) kısımlarını da yazabilirsiniz, ";"den sonrasını AsmPro dikkate almayacaktır.

Yazmayı bitirdiğinizde tekrar ESC tuşu ile editörden çıkın. `A` (tek harf, sadece A) komutunu yazıp enter'a bastığınızda derleme işlemi gerçekleşecektir. Ekranda aşağıdaki gibi satırlar görmelisiniz:

```
>a
Pass 1..
Pass 2..
No Errors
>
```

Derlediğiniz kodu çalıştırmak için sadece J komutunu (jump)

vermeniz yeterlidir. Aynı şekilde “j start” da diyebilirsiniz, bu şekilde kodunuzun belirli bir yerinden itibaren çalıştırmaya başlayabilirsiniz.

Eğer derleme işlemi sırasında hatalar görüyorsanız, konsol ekranında (editör dışında) sağ tuş ile Assembler/Preferences/Assembler menü seçeneğini seçerek derleyici ayarlarına ulaşabilirsiniz. Burada “UCase = LCase” ve “Label :” seçeneklerini seçin. Bunlar sırasıyla büyük/küçük harf farkını kapatır ve etiketlerden sonra “:” konma zorunluluğu getirir.

Eğer hata almaya devam ediyorsanız yazdığınız kodda hata vardır. İyice gözden geçirip tekrar deneyin.

J komutu ile programınızı çalıştırdığınızda aşağıdakine benzer satırlar göreceksiniz;

```
D0: 00000285 00000000 00000000 00000000
A0: 11390FC1 00000000 00000000 00000000
SSP: 110BCBF4 USP: ...
```

Bu gördükleriniz 68000’in register’larıdır. D0 register’ının değeri ‘plazma’nın harflerinin hafızadaki değerlerinin toplamı olan $285 olarak görülmekte. Bu şekilde programımız bittiğinde register'ları hangi durumda bıraktığını görebilirsiniz.

Ayrıca aşağıdaki komutlarla 'checksum' adresindeki değeri de görebilirsiniz,

```
>h checksum
```

Bu komut ile hafızanın dump’ını alırken, aşağıdaki komutla hafızanın disassemble edilmiş halini görebilirsiniz. Yön tuşlarıyla hafızanın başka bölgelerini de görebilirsiniz.

```
>d checksum
```

Benzer şekilde a ve d komutlarını arka arkaya verdiğinizde programınızın derlenmiş halini görebilirsiniz.

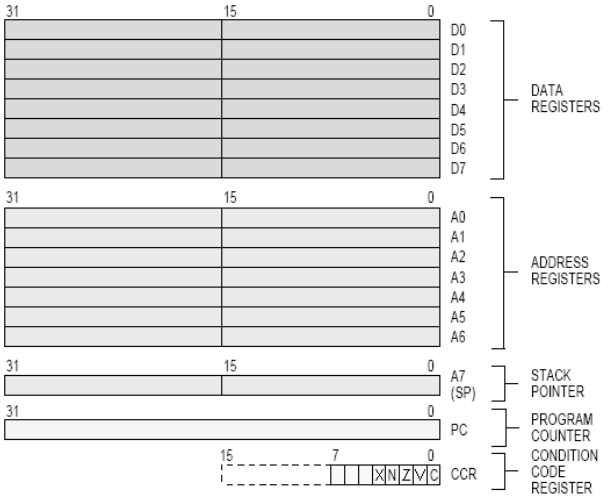
Programımızda kullandığımız 68000 komutlarının anlamlarını öğrenmeye başlamadan önce 68000 işlemcinin register’larına kısaca bir göz atalım.

68000 Register’ları

68000 işlemci 16/32 bitlik bir işlemcidir, 32 bitlik veriler üzerinde çalışabilir ancak veri yolu 16 bit, adres yolu 24 bitliktir.

68000 işlemcilerde 8 adet data register’ı, 8 adet adres register’ı, durum register’ı (SR, status register), 2 adet yığın göstergeci (USP ve SSP) ve program göstergeci vardır.

Registerlarla .b, .w, .l gibi sonekler (suffix) kullanarak sırasıyla byte (8 bit), word (16 bit) ve longword (32 bit) işlem yapabilirsiniz. Buna ileride tekrar değineceğiz.



Şekil 2.

Data register’ları D0-D7 olarak yazılır. Aritmetik işlemler ve diğer depolama gereksinimleri için kullanılabilirler. Bu register’lar üzerinde bayt (8 bit), word (16 bit) veya longword (32 bit) olarak işlem yapılabilir.

Adres register’ları A0-A7 şeklindedir. Belirli hafıza adreslerini tutarlar ve o adreslerle ilgili okuma/yazma işlemlerini gerçekleştirmede kullanılırlar (pointers). Data register’ları gibi 32 bitlik register’lardır. Ancak bu register’lar üzerinde sadece word ve longword işlem uygulanabilir. (Örneğin ADD.W #1,A0 kullanılabilir, ancak ADD.B #1,A0 kullanamazsınız.)

Bu noktada ufak bir ek bilgi verelim, data register’larının tümünü dilediğiniz gibi kullanabilirsiniz, ancak adres register’larının sonuncusu olan A7 aynı zamanda yığın göstergecidir (SP, stack pointer) Bu nedenle bu register’ı diğerleri gibi kullanmayın!

Durum register’ı (SR, status register) 2 bayt’tır ve aşağıdaki bilgileri içerir;

1. 0: C,Carry: Aritmetik ve kaydırma komutlarından etkilenir
2. 1: V,Overflow: Taşma olduğunda set edilir
3. 2: Z,Zero: Son işlemin sonucu sıfır ise set olur
4. 3: N,Negative: Son işlemin sonucu negatif ise set olur
5. 4: X,Extended: Aritmetik işlemlerden etkilenir
6. 5-7: Kullanılmıyor
7. 8: I0: Interrupt bitleri
8. 9: I1: Interrupt bitleri
9. 10: I2: Interrupt bitleri
10. 11: Kullanılmıyor
11. 12: Kullanılmıyor

12. 13: S,Supervisor: İşlemcinin hangi modda çalıştığını gösterir
13. 14: Kullanılmıyor
14. 15: T,Trace: İşlemci trace modunda ise set durumdadır

SR'nin ilk bayt'ına kullanıcı bayt'ı, ikinciye ise sistem baytı adı verilir.

Komutlara Genel Bakış

68000 işlemci komutlarını veri transfer, aritmetik, lojik ve dallanma komutları olarak gruplandırabiliriz. Şimdi bu komutların bir listesini verelim:

Not: Bu liste 68000 komutlarının tam listesi değildir, en sık karşılaşacağımız ve sık kullanılan komutlardır, bunlar dışındaki komutlara pek ihtiyacınız olmayacaktır, yine de yeri geldikçe onlara da değineceğiz.

Listede kullanılacak semboller aşağıdaki gibidir:

label: Etiket veya adres

reg: Register

an: Adres register'ı, A0-A7

dn: Data register'ı, D0-D7

kaynak

hedef

<ea>: Efektif adres

#n: Değer

Dallanma Komutları

1. *BCC label* Koşullu dallanma
2. *BRA label* Koşulsuz dallanma (JMP)
3. *BSR label* Alt rutine dallanma (RTS ile dönülür)
4. *JMP label* Koşulsuz dallanma (BRA)
5. *JSR label* Alt rutine dallanma (RTS ile dönülür)
6. *RTS* Alt rutinden geri dön (JMP, BSR)

Aritmetik İşlem Komutları

1. *ADD kaynak,hedef* hedefe, kaynağı ekler
2. *ADDI #n,<ea>* adrese n ekler
3. *CLR <ea>* Adres temizlenir
4. *CMP kaynak,hedef* Kaynak ve hedef adreslerin bitleri karşılaştırılır
5. *DIVS kaynak,hedef* Hedef (32 bit) Kaynağa (16 bit) bölünür, bölme işleminin sonucu Hedef'in alt-word'ünde, kalan ise üst-word'de tutulur. Bölme işlemi işaret dikkate alınarak yapılır (Signed)
6. *DIVU kaynak,hedef* Hedef (32 bit) Kaynağa (16 bit) bölünür, bölme işleminin sonucu Hedef'in alt-word'ünde, kalan ise üst-word'de tutulur. Bölme işlemi işaret dikkate alınmadan yapılır (Unsigned)
7. *MULS kaynak,hedef* Hedef (word), kaynak (word) ile çarpılır, sonuç hedef'te tutulur (longword)
8. *MULU kaynak,hedef* Hedef (word), kaynak (word) ile çarpılır, sonuç hedef'te tutulur (Unsigned)
9. *NEG <ea>* Negatife çevirir (2'lik sistemde komplement)
10. *SUB kaynak,hedef* hedef'ten kaynağı çıkarır
11. *SUBI #n,<ea>* adresten n çıkarır
12. *TST <ea>* Adres test edilir sonuç N ve Z bayraklarını etkiler

Mantıksal İşlem Komutları

1. *AND kaynak,hedef* hedef, kaynak ile mantıksal VE işleminden geçer
2. *ANDI #n,<ea>* hedef, n ile mantıksal VE işleminden geçer
3. *EOR kaynak,hedef* hedef, kaynak ile mantıksal XOR işleminden geçer
4. *EOR #n,<ea>* hedef, n ile mantıksal XOR işleminden geçer
5. *NOT <ea>* Adresin mantıksal DEĞİL'i alınır
6. *OR kaynak,hedef* hedef, kaynak ile mantıksal VEYA işleminden geçer
7. *ORI #n,<ea>* hedef, n ile mantıksal VEYA işleminden geçer

Bit İşlem Komutları

1. *BCHG #n,<ea>* Adresin n'inci bitini ters çevirir (toggle)
2. *BCLR #n,<ea>* Adresin n'inci bitini temizler (sıfır yapar)
3. *BSET #n,<ea>* Adresin n'inci bitini set eder (bir yapar)
4. *BTST #n,<ea>* Adresin n'inci bitini test eder, Z bayrağını etkiler
5. *ASL, ASRn, <ea>* Aritmetik sola, sağa kaydırma
6. *LSL, LSRn, <ea>* Mantıksal sola, sağa kaydırma
7. *ROL, RORn, <ea>* Sola, sağa döndürme

Veri transfer Komutları

1. *EXG rn,rn* İki register'ın içerikleri değiştirilir
2. *LEA <ea>,an* an adres register'ına verilen efektif adres yüklenir
3. *MOVE kaynak,hedef* Kaynak'tan okunan değer hedef'e yazılır
4. *SWAP dn* dn data register'ının alt-word'ü ile üst-word'ünü yer değiştirir

Genel olarak 68000 komutları bunlardır, bu komutların pek çoğunda .b, .w ve .l gibi işlemin kaç bit üzerinden yapılacağını belirleyebiliriz.

Bu sayıda veri transfer komutlarına biraz daha detaylı bakalım, bu diğer komutları da anlamanıza yardımcı olacaktır. Sonraki yazılarımızda diğer komutları da detaylıca inceleyeceğiz.

Veri Transfer Komutları

Move komutları veri transfer işlemini gerçekleştirirler. İleride detaylı olarak değineceğimiz pek çok adresleme yöntemini kullanabilirler.

Genel kullanımı "MOVE kaynak,hefef" şeklindedir; aşağıdaki örnekleri inceleyiniz;

```
move.l #12,d0    ;D0 registerine
                   ;12 degeri koy (32bit)

move.w #$1234,d7  ;D7'ye $1234
                   ;degeri koy (word)
```

Bir register'a word olarak değer girdiğinizde 32 bitlik register'ın alt 16 biti değişir ve üst 16 bit aynı kalır. Örneğin;

```
move.l #$11223344,d0   ;D0 => $11223344
move.w #$5566,d0       ;D0 => $11225566
move.b #$77,d0         ;D0 => $11225577
```

NOT: Komutlarda sonek kullanmadığınızda .w olarak kabul edilir. Ancak bazı komutlar sadece 8 bit veya sadece 16 bit destekliyorsa derleyici uygun şekilde derleme yapacaktır.

Şekil 3.

MOVE komutunda kaynak ve hedef birer register olabileceği gibi bir hafıza adresi de (ya da etiket) olabilir. Aşağıdaki örnekleri inceleyin;

```
move.l src1,d0   ;D0 => $00000001
move.l src1,dst   ;dst[0] => $00000001
move.w src2,d    ;D5 => $XXXX00AA
move.l src2,6    ;D6 => $00AA0055
rts

src1: dc.l $01,$02,$03,$04
src2: dc.w $AA,$55
dst:  dc.l $00,$00,$00,$00
```

Örneklerde verilen src1, src2 ve dst'nin hafızada nasıl durduklarına bakmak anlamamıza yardımcı olacaktır:

```
src1: $00000001,$00000002,$00000003,$00000004

src2: $00AA,$0055 ;$00AA0055 gibi
                    ;(longword) dü#ünebiliriz.

dst: $00000000,$00000000,$00000000,$00000000
```

Move komutunun başka bazı kullanım şekilleri de vardır, bunlara ileride tekrar bakacağız. Diğer Veri transfer işlemi yapan komutlar aşağıdadır;

```
lea <adres>,a0 ;<adres> degerini
                  ;A0 register'#na koy
```

Burada dikkat edilmesi gereken nokta, LEA (Load Effective Address) komutu <adres>'in gösterdiği yerdeki değeri değil adresin kendisini register'a yazacaktır. Eğer adresin gösterdiği (point ettiği) değeri okumak istiyorsanız MOVE <adres>,<register> kullanmalısınız.

```
lea $11223344,a0 ;A0 => $11223344

lea src,a1      ;A1 => $1CF6723F (farkli olabilir)
                ;"plazma" yazisinin baslangic
                ;adresi

lea src(PC),a1  ;A1 => $1CF6723F (farkli olabilir)
                ;"plazma" yazisinin baslangic
                ;adresi

src:
    dc.b 'plazma',0
```

Son örnekte src'nin adresi Program Sayacı (PC)'ye göre relatif olarak okunur ve A6'ya yazılır. Bunun bir öncekinden farkı hafızada longword olarak değil word olarak tutulmasıdır. Yani "bulunduğum yerden XXXX ilerisi" anlamına gelir.

Tahmin edebileceğiniz gibi aşağıdaki 2 kullanım aynı sonucu verecektir:

```
move.l  #src,a6
lea     src,a6
```

Veri transfer komutlarına aşağıdaki 2 komutu da ekleyebiliriz;

```
exg d0,d1;D0 <=> D1 (degerleri degistirir)
exg a0,d2;A0 <=> D2 (degerleri degistirir)
```

NOT: exg (exchange) komutu sadece 32 bit destekler, bu durumda exg veya exg.l yazabilirsiniz. Ancak exg.w ve exg.b kullanamazsınız.

```
swapd0  ;D0'#n alt 16 bit ile üst
        ;16 bitinin yerini de#i#tirir
```

NOT: swap komutu sadece data register'ları ile çalışır. Adres register'ları ile kullanamazsınız.

Yazımı bitirmeden önce değinmek istediğim son bir konu daha var.

Amiga'nın Hafıza Yapısı

Standart bir Amiga 500'te 512 KB Ram vardır. Bu alan $000000 - $7FFFFF arasındadır. Amiga'nın işletim sistemi tümüyle multi tasking olduğundan diskten yüklenen bir program hafızanın (Chip Ram veya varsa Fast Ram) herhangi bir yerine otomatik olarak yerleştirilir ve orada çalıştırılır. Bu nedenle, pek çok eski Amiga programcısının yaptığı gibi, mutlak adres (absolute memory) kullanımından kaçınmak gerekir. Kaldı ki AmigaDOS programınızın yüklenmesi sırasında tüm relocate işlemlerini sizin yerinize otomatik olarak yapacaktır.

Hafızanın genel görünümü aşağıdaki gibidir:

1. $000000-$07FFFF Chip RAM
2. $080000-$1FFFFF Rezerve
3. $200000-$9FFFFF Fast RAM (eğer varsa)
4. $A00000-$BEFFFF Rezerve
5. $BFD000-$BFDF00 CIA B (Çift adresler)
6. $BFE001-$BFEF00 CIA A (Tek adresler)
7. $C00000-$DFEFFF Rezerve (Genişleme için)
8. $DFF000-$DFFFFF Custom Chip (Hardware) Register'ları
9. $E00000-$E7FFFF Rezerve
10. $E80000-$EFFFFF Genişleme Portları
11. $F00000-$F7FFFF Rezerve
12. $F80000-$FFFFFF ROM

Amiga'da kod yazabilmek için hafıza yapısını bilmek kadar Amiga'nın özel çiplerini (custom chips) de bilmek gerekir. Bunlar $DFF000-$DFFFFF arasında bulunan hardware register'ları tarafından kontrol edilen, mikro işlemciden bağımsız çalışabilen, Amiga'nın görüntü ve ses çıkışını kontrol edebilen özel çiplerdir.

Agnus, Denice ve Paula olarak adlandırılan bu özel çipler Amiga'nın, döneminin en güçlü bilgisayarı olmasındaki temel faktörlerdir.

Agnus, blok halindeki hafızanın bir yerden başka bir yere taşınması işlemini (blitting) gerçekleştirir. Bu işlem sırasında aynı zamanda taşınan veri üzerinde çeşitli işlemler (masking, AND, OR) uygulayabilir. Ek olarak çizgi çizebilir ve bir poligonun içini doldurabilir. Denice, ekran görüntüsü ile ilgili tüm işleri üstlenmiştir (copper). Paula ise tüm giriş/çıkış işlemlerinden sorumludur. (Disk sürücü ve ses gibi.) Custom Chip'ler ve hardware register'lara ileriki sayılarda çok daha detaylı olarak değineceğiz.

Bu noktada Fast Ram ve Chip Ram arasındaki farka kısaca değinelim. Chip Ram, 68000'in ve Amiga'nın özel çiplerinin doğrudan erişebileceği bellektir. Fast Ram ise sadece 68000 tarafından erişilebilir. Burada önemli bir nokta; ses ve görüntü ile ilgili tüm verilerin Chip Ram'de durması gerektiğidir.

Özet

Amiga Assembly'ye giriş niteliğindeki bu yazımıza burada son veriyoruz. Bu sayıda Amiga'nın register'ları ve data transfer komutlarına değindik, ayrıca kodlarımızı nasıl derleyeceğimizi öğrendik.

Önümüzdeki sayıda yeni örnek kodlarımızla 68000 komutlarını incelemeye devam edeceğiz. Ek olarak Amiga Assembly ile kod yazarken olmazsa olmaz konulardan olan kütüphane (library) fonksiyonlarının nasıl kullanıldığını da göreceğiz.

Ayrıca AsmPro'nun nasıl kullanıldığı ile ilgili ayrı bir yazıyı da bir sonraki sayıda bulabilirsiniz.

Soru ve önerileriniz için bana aşağıdaki adresten ulaşabilirsiniz. Bir sonraki sayıya kadar hoşçakalın.

semseddinm (at) gmail (nokta) com

Yazılım Doğrulamaya Giriş

Bilgem 'Nightlord' Çakır

Bu yazımızda son yıllarda giderek önemi artan yazılım doğrulama konusuna giriş yapacağız. Yazılım testi konusundaki temel problemlerden ve tekniklerden bahsedeceğiz.

Yazılım Doğrulama nedir.

Bu konuda pek çok değişik tanım bulmak mümkün. Bazı tanımlar işin teknik tarafına yoğunlaşırken, kimi tanımlar ise konunun yönetimsel tarafına değiniyor. İki örnek tanıma bakalım.

Yazılım Doğrulama bir programın spesifikasyonlarında anlatıldığı şekilde davranıp davranmadığını sınama aktivitesidir.

Yazılım Doğrulama, bir yazılım projesinin zamanında ve bütçe dahilinde, ölçülebilir olarak hedeflenen spesifikasyonlara doğru ilerlemesini sağlayan aktiviteler bütünüdür.

Her iki tanımda da ortak olan bazı öğeler olduğunu görebilirsiniz. Dikkatinizi çekmesi gereken belki de en önemli nokta test edilen programın bir spesifikasyonu olması gerektiği. Yani doğrulamayı yapan kişilerin programı neye göre doğruladıklarını bilmeleri gerekir. Programın ne yapmayı vaad ettiğini bilmeden, vaad ettiği şeyi yapıp yapmadığını doğrulamak mümkün değildir. Spesifikasyonlara bundan sonra kısaca Spek diyeceğiz.

Doğrulama Türleri

Yazılım doğrulama çabası farklı türlerde testlerin yapılmasıyla yürütülür. Değişik amaçları olan farklı türlerde testler programınızın farklı yönlerden doğrulanmasına yardımcı olur ve kalitesinin artmasını sağlar

Kapalı Kutu Testleri

Eğer yazdığınız yazılımı kapalı bir kutu olarak düşünür ve sadece Spekte belirtilen davranışları test etmeye çalışırsanız kapalı kutu testleri yapıyorsunuz demektir. Genelde kapalı kutu testleri, sadece test edilen yazılımın arayüzünü kullanarak sisteme çeşitli girdiler verip, sistemin beklenen şekilde davranıp davranmadığına bakarlar. Bu arayüz bir GUI programında penceredeki düğmeler, text alanları vs olabilir. Bir konsol programında verilen opsiyonlar olabilir. Bir kütüphane için API olabilir. Eğer programınız harddiskten bir data dosyası kullanıyorsa o dosya olabilir. Veya Network haberleşmesi yapan bir programa gelen network paketleri de "arayüz"deki girdiler olabilir. Kapalı kutu testi yaparken hep böyle girdiler hazırlayıp sisteme verir, ve sistemin verdiği cevaba bakarız. Bu girdileri hazırlarken nelere dikkat etmemiz gerektiğine birazdan değineceğiz.

Şeffaf kutu testleri

Çoğu zaman sadece kapalı kutu testleri yapmak yeterli olmayacaktır. Özellikle programınız dar bir arayüze sahip olmasına karşın çok karmaşık işlemler yapıyorsa (ki çoğu zaman programlarımızı tasarlarken böyle olmasını hedefleriz) mutlaka şeffaf kutu testleri de yapmak gerekir. Bu testleri hazırlarken sisteminizin içinde yer alan algoritmaları saklanan bilgileri vs dikkate alarak hareket edersiniz. Bunun nasıl olduğunu da birazdan anlatacağız.

Zorlayıcı Testler

Özellikle projeniz bir miktar ilerledikten sonra, yazdığınız programın ağır yükler altında nasıl davrandığını görmek isteyebilirsiniz. Örneğin network üzerinden çalışan bir sunucunun, çok fazla sayıda istemci olduğunda nasıl davrandığını test etmek, veya bir data dosyasını inceleyen programınıza devasa büyüklükte dosyalar vermek, bu sınıfa giren testlerdir. Bu yazımızda bu testlere değinmeyeceğiz.

Performans testleri

Çoğu zaman speklere programınızın performansı ile ilgili vaatler konabilir. Örneğin programın yakalamayı hedeflediği frame rate veya kullanıcının düğmeye basması ile ekrana menünün çıkması arasındaki maksimum süre gibi hedefler olabilir. Bazı sistemlerde bunlar hayati önem de taşıyabilir. Bu yüzden programın performans hedeflerini tutturabildiğini doğrulamak için çeşitli testler yapılabilir. Bu testler genellikle sistemde bulunan çeşitli zamanlayıcıları kullanarak, iki olay arasındaki zaman farkını ölçüp istenen değerin altında kalıp kalmadığını kontrol ederler.

Hataların geri gelmesini önleme testleri

Çok önemli bir test grubu da budur. Genelde programın geliştirme aşamalarında pekçok hata ortaya çıkar. Bulunan hatalar geliştiriciler tarafından giderilmeye çalışılır. Fakat çok sık yaşanan bir problem, bazı hataların düzeltilmesine rağmen tekrar tekrar su yüzüne çıkmasıdır.

Bunun sebebi genellikle, bir geliştiricinin bir hatayı düzeltirken, daha önce yapılan bir düzeltmeyi bozabilmesi, ve eski bugların yeniden ortaya çıkmasına sebep olmasıdır.

Bu yüzden, genellikle düzeltilen her hata için, o düzeltmeyi doğrulayan bir test hazırlanır. Ardından bu test, böyle diğer bütün testlerle beraber grup olarak periyodik aralıklarla çalıştırılır. Genellikle bir grup testin her gün çalıştırılması otomatize edilir. Böylece eğer bir geliştirici kodda yanlışlıkla eski bir hatayı geri getirecek bir değişiklik yaparsa, bu durum hemen bir gün içinde otomatik testlerce yakalanır. Böylece projenin hep ileri doğru gitmesi sağlanır.

Veri Bozma Testleri

Son olarak değineceğimiz test türü de veri bozma testleri. Bu testlerle, genelde bir programın girdi olarak aldığı veriler rast-

gele veya belli kurallara bağlı olarak bilinçli olarak bozulur ve sisteme verilir. Sistemin bu koşullarda, bozuk veriyi zarif bir şekilde (yani göçmeden) reddetmesi veya bazı küçük hataları düzeltebilmesi isteniyor olabilir. Bu konuda beklentiler her ne ise (bu beklentiler de spekte olmalı), bu beklentilerin gerçekleşip gerçekleşmediği de bu testlerle sınanabilir.

Yazılım Doğrulama Birimleri: Test Birimleri

Buraya kadar anlattığımız bütün test türlerinde genellikle testler küçük veri gruplarından oluşur. Bir seferde programa verilecek veriler ve programın vermesini beklediğimiz cevap bir "Test Birimi"ni oluşturur.

Örneğin elinizde toplama yapan bir program var. Bu programa iki sayı veriyorsunuz ve o da toplamını döndürüyor. Bu programa 3 ve 5 verip cevap olarak 8 beklemek bir test birimidir.

İyi test birimleri tasarlamak çok önemli bir yetidir. Her programın veya kütüphanenin test birim tasarımı farklı olacaktır. Fakat test birimleri tasarlarken dikkat edilecek bazı ortak noktalar var. Bunlara değinelim.

Kapalı kutu test tasarımı

Kapalı kutu test tasarımı yaparken bildiğiniz (veya bildiğinizi varsaydığınız) tek şey test ettiğiniz yazılımın spekidir. Spekin parçası olarak programla nasıl etkileşileceği (yani arayüz) açıklanmıştır. Mesela daha önce değindiğimiz toplama örneğini ele alalım.

Toplama yazılımı: Bu yazılım iki adet iki basamaklı pozitif sayıyı girdi olarak alır, bu sayıların toplamını çıktı olarak döndürür. Eğer hatalı bir girdi verilirse sonuç -1 olarak döndürülür.

Bu örnekte bahsettiğimiz yazılım bir gui programı da olabilir, bir konsol programı da, ya da bir APInin parçası olan bir fonksiyon da olabilir. Biz burada bunu önemsemiyoruz. Yani girdileri, GUI veya konsola mı veriyoruz, yoksa bir fonksiyona arguman olarak mı geçiriyoruz, buna şimdi dikkat etmeyeceğiz. Biz daha çok test birimini belirlemeye çalışacağız. Yani bu yazılımı hangi sayı çiftleri ile test edeceğimizi tartışacağız.

Sınır Koşulları

Bir programa verilen olası girdi değerlerinin tümünü deneyebilseydik programla illgili kendimizi çok güvende hissedebilirdik. Fakat bu genellikle imkansızdır. Bu yüzden bir Test Birimi tasarlarken mutlaka girdilerin alabileceği hangi değerlerin kritik olduğuna ve olası hataları ortaya çıkarabileceğine bakarız.

Genel olarak programlar kabul ettikleri girdiler ile ilgili bazı sınırlar belirtirler. Bu sınırların dışında kalan değerlerin kabul edilmediğini veya bir sınırın üstünde ve altında değerler için farklı işlemler yapıldığını belirtebilirler.

Genellikle pekçok hata bu sınırlar ve onların hemen yakınında olur. Mesela örnek programımızda verilen sayıların iki basamaklı pozitif sayılar olması bekleniyor. Programın içinde sayının üst sınırının kontrol edildiğini düşünelim. Bu kontrolu yapan kod şöyle birşey olabilir.

```
if ( Verilen_Sayi < 99 )
{
    ...
```

veya belkide şöyledir:

```
if ( Verilen_Sayi >= 100 )
{
     ...
```

ya da böyle:

```
if ( Verilen_Sayi < 100 )
{
    ...
```

Dikkat ederseniz buradaki karşılaştırma koşulu bir olası hata noktası. Verdiğimiz ilk iki kod hatalı. İlki geçerli bir girdi olan 99 değerini reddedecek. İkincisi ise geçersiz bir girdi olan 100 değerini kabul edecek.

Böyle problemlere çok sık rastlandığı için genelde sınır değerleri yakınında ve üzerinde değerler içeren test birimleri hazırlamak mantıklıdır. Örneğin;

- -1, 5
- 0, 5
- 1, 5
- 98, 5
- 99, 5
- 100, 5
- 12, -1
- 12, 0
- 12, 1
- 12, 98
- 12, 99
- 12, 100

Bu 12 test birimi ile sınır koşullarını her iki girdi için de test etmiş oluyoruz. Eğer bütün olasılıkları test edecek olsaydık 100 x 100 toplam 10000 test birimi hazırlamak gerekirdi. Burada sadece 12 test birimi ile 10000 birimlik testimize çok yakın bir güven elde edebiliriz (Bu güveni nasıl nümerik olarak ölçebiliriz birazdan göreceğiz)

Denklik Sınıfları

Bir programa giren girdiler, eğer sürekli bir kümeden gelmiyorsa, sınırlardan bahsetmek mümkün olmayabilir. Örneğin haftanın günlerinden biri girdilerden birini oluşturuyor ise, ve program hafta sonu için farklı hafta içi için farklı davranacağını söylüyorsa sınırlar nedir. Cuma olduğunu düşünebilirsiniz. Belki... Ama bu tip durumlar (ve sürekli kümeler için de) sınır koşullarına bakmak dışında bir perspektif daha vardır. Girdinin alabileceği değerler kümesini "denklik sınıfları"na bölmek.

Denklik sınıfları, adından da tahmin edebileceğiniz gibi bir girdinin alabileceği bütün değerleri speklere göre sınıflara ayırır. Aynı sınıfa ait üyelerin ayni şekilde işlem göreceği ve bir denklik sınıfından sadece bir (veya birkaç) değerin testte kullanılması öngörülür. Bu şekilde tıpkı sınır koşullarını kullandığımız zamanki gibi test birimi sayısını azaltabiliriz.

Kod Kapsama ve Şeffaf kutu testleri

Az önce programa ve testlerimizin programın hatalarını ne kadar yakaladığına dair güvenimizi nasıl ölçebileceğimizi sorgulamıştık. Bu kaygıyı birebir cevaplamamakla beraber, bazı sayısal metrikler mevcuttur ve sayısal metrikler her zaman "içimden bir ses böyle söylüyor"dan daha iyidir.

Fakat ne zaman ki kod kapsamadan bahsediyoruz, o zaman kapalı kutudan bahsetmiyoruz demektir. Çünkü birazdan göreceğiniz gibi, kod kapsama tekniklerinden faydalanmak, yazılan kodun içini bilmeyi ve bu bilgiden faydalanmayı gerektirir. Bu bilgiyi kullanarak yapılan testlere de "Şeffaf Kutu Testleri" denir.

Aşağıdaki fonksiyona bir bakın:

```
int Topla( int Sayi1, int Sayi2)
{
    if ((Sayi1>=0)&&(Sayi1<100)){
        if((Sayi2>=0)&&(Sayi2<100)){
            return(Sayi1+Sayi2);
        }else{
            //Sayi 2 istenilen aralikta degil
            return -1;
        }
    }else{
        // Sayi 1 istenilen aralikta degil
        return -1
    }
}
```

Bu fonksiyonu test ederken ne zaman emin oluruz. İşte kod kapsama kavramı burada devreye girer. Kod kapsama, yazılan bir kodun içindeki değişik birimlerden hangilerinin testler esnasında sınandığını ve ne kadarının sınanmadan geçtiğini ifade eder. Değişik kod kapsama türleri vardır. Şimdi bunlardan bir kaçına değinelim.

Kod satırlarını veya kod dallarını kapsamak

Bu örneğimizi bir adet test birimi ile test ettiğimiz zaman hangi satırların çalıştığına bakabiliriz. Koşullu ifadelerden (yani if-else) dolayı bir test birimi bu fonksiyondaki her satıra uğramayacaktır.

Bir grup test birimini çalıştırdıktan sonra fonksiyonda uğranan satırların sayısını daha da artırmış oluruz. Böylece iyi seçilmiş test birimleri ile her hangi bir programın sahip olduğu kod satırlarının büyük bölümünün "kapsanmasını" sağlayabiliriz. Ve diyebiliriz ki bu programımızı testlerimizle %85 satır kapsaması sağlayarak test ettik. Bu ölçülebilir bir metriktir. Örneğin koda eklemeler yaptıkça kapsama yüzdeniz azalacaktır. Bir proje ekibi kapsama yüzdesi örneğin %70'in altına düşen bir yazılımı çıkarmayı reddedebilir. Çünkü kodun %30unun test edilmediği ve orada kritik buglar olabileceğini düşünebilirler.

Bazen de koddaki satırların kapsamasını incelemek yerine koddaki dalların (blokların) kapsamasını inceleyebiliriz. Bunu takip etmek kimi projelerde daha mantıklı olabilir. Fakat temelde aynı mantıktır.

Son olarak Kapsama değerlerini ölçen programlar vardır. Yani kapsama değerlerini siz elle ölçmezsiniz. Bunun için favori derleyicinizin kod kapsama (code coverage) özelliklerini araştırın

Koşul ifadelerini kapsamak

Daha önce en çok hata yapılan yerlerin koşullar olduğundan bahsetmiştik. Eğer karmaşık koşul ifadeleri yazılırsa, dikkat etmemiz gereken bir nokta daha var. Şu örneğe bir bakın. Eğer elimizde şöyle bir if satırı olsaydı:

```
if ( ( Sayi1 >= 0 ) && ( Sayi2 < 99 ) )
```

Bu if satırından dolayı kod iki farklı yere atlayabilir. Ya bir alt satıra ya da else bölümüne. Diyelim ki Sayı1 ve Sayı2 için sırasıyla -1,99 değerlerini verdik. Bu durumda aradaki && işleminin yarattığı bir problem var. Bu operator kendisinden önceki ilk koşula (Sayı1 >= 0) bakıp onun yanlış olduğunu gördüğü anda else bölümüne atlar. Yani ikinci koşulu test etmez. Bu yüzden biz ikinci koşulda yaptığımız hatayı farketmeyebiliriz.

Bu yüzden kimi projeler metrik olarak "Koşul Kapsaması"nı da kullanırlar. Test birimlerinin hangi koşul ifadelerine uğradığını takip ederler. Kod satırı kapsamadan farklı olarak genelde koşul kapsama metrikleri kullanıldıkları yerlerde %100 koşul kapsaması aranır.

Durum Motorlarını kapsamak

Bir diğer şeffaf kutu yaklaşımı da durum motorlarını test ederken karşımıza çıkar. Bir yazılım eğer çeşitli durumlara ve durumlar arası tanımlı geçişlere sahip bir durum motoru olarak tasarlanmış ise, bu durumda test birimlerimizin hangi durumları ve hangi geçişleri kapsadığından bahsedebiliriz. Durum motorlarını test ederken ayrıca komutların gonderiliş sıraları da karıştırılarak durum motorunun tepkisi test edilir. Bu konuyu bu yazıda daha detaylı incelemeyeceğiz

Hata Enjeksiyonu

Diyelim ki kapalı kutu testleriyle başladınız. Speklere baktınız ve görebildiğiniz sınır koşulları ve denklik sınıflarını kullanarak 50 tane test birimi hazırladınız. Sonra bu testlerin sağladığı kod kapsamasını ölçtünüz ve %65 çıktı. Ardından şeffaf kutu yaklaşımına geçtiniz. Uğranmayan kod dallarını tespit edip onların bir kısmına uğranmasını sağlayacak şekilde 10 tane daha test birimi eklediniz. Tekrar kapsamayı ölçtünüz ve %90 çıktı. Biraz inceleyince neden bazı kod dallarına hiç uğramadığınızı gördünüz. Mesela şöyle kodlar var.

```
cSoldier oSoldier = new Soldier( kBlue );
if ( oSoldier == NULL ){
    std::cout << "bellek bitti"
              << std::endl;
    exit 1;
}
```

veya;

```
XYZLibraryError_t oError = XYZLibraryInit();
if ( oError != XYZ_NO_ERROR ){
    std::cout << "XYZ kutuphanesini baslatamadim"
              << std::endl;
    exit 1;
}
```

bu tip if'lerin içine giremiyorsunuz, çünkü bazı ekstra hatalar olması gerekiyor. Sizin testlerinizde doğal olarak makinenizin belleği normal şartlarda bitmiyor ve XYZ kütüphanesi hep başarıyla başlayabiliyor.

Bu noktada mevcut kapsama değerleriyle yetinip daha fazla test birimi üretmeyebilirsiniz. Bu çoğu zaman mantıklı bir karardır. Bazı projelerde bundan fazlası gerekebilir. Mesela Aya İniş Modülünün yazılımını yazıyorsanız, ve oradaki astronota "pardon ya bellek bitti ben çıkıyorum" diyemeyeceğiniz için, bu tip hata koşullarının da test edilmesi gerekebilir. İşte bu test yaklaşımına da "Hata Enjeksiyonu" deniyor.

Hata enjeksiyonu yapmak değişik şekillerde mümkün olabilir. Kütüphaneler yerine aynı arayüze sahip ve sizin istediğiniz hataları döndüren taklit fonksiyonlar (bunlara stub denir) yazabilirsiniz. Ya da test ettiğiniz sistemde bir takım harici araçlarla hata koşullarını yaratabilirsiniz.

Sonuç

Yazılım doğrulama alanı yazılım geliştirmeye göre daha yeni ve fazla çözülmemiş bir alandır. Fakat yine de yazılım geliştirme projelerinde kaliteyi ve ilerlemeyi ölçen önemli bir metrik sağlar. Yazılım doğrulamanın alt alanlarından kısaca bahsettiğimiz ve Test Birimi tasarlamanın temellerine giriş yaptığımız bu yazı umarız ileriki projelerinizde faydalı olur.

nightlord (at) nightnetwork (nokta) org

Yararlanilan Kaynaklar:

- Testing Computer Software 2nd Ed.; Kaner, Falk, Nguyen
- How to Break Software: A Practical Guide to Testing; Whittaker
- Ship it! A Practical Guide to Successful Software Projects; Richardson, Gwaltney

İpuçları

Emir 'Skate' Akaydın

Commodore 64

Şekil 1.

$F5C1 (62913) adresinin en son yükletilen dosyayı ekrana yazdığını biliyor muydunuz?

```
load"program",8,1
sys 62913
```

Eğer dosyayı load"*",8,1 şeklinde yüklediyseniz ekrana * karakteri yazılacaktır. Yani bu komut sizin yüklediğiniz en son dosya isminin tutulduğu tampon belleği ekrana yazar. Peki nerede bu tampon bellek? *$a000* adresi Basic ROM'unun başladığı adrestir. Yüketilen dosya isminin uzunluğuna bağlı olarak bu bellek alanı $9fff'den geriye doğru kullanılır, yani Basic ROM'unun başlangıcından geriye doğru. Örneğin dosyayı "*" ile yüklemişsek bellek başlangıcı tam olarak $9fff'de yer alır. Ama "dutch breeze" diye bir dosya yüklediysek, dosya ismi 12 karakter olduğu için $a000-12 = $9ff4 adresinden itibaren hafızaya yerleşecektir. Bu bellek adresinin başlangıcı ise zeropage'de $bb ve $bc adreslerinde 16 bit şeklinde durur. Örneğin "dutch breeze" için $bb = $f4, $bc = $9f gibi.

6510 ASM'de hiçbir flag'i (zero flag, carry flag v.b.) etkilemeyen komutlar nelerdir?

NOP, JSR, JMP, PHA, PHP, RTS, STA, STX, STY. Bu komutlar haricindeki tüm 6510 assembler komutları en az bir flagi etkiler.

$D011'e sıfır değeri verdiğimizde DMA satırlarından kurtuluyoruz ve iç ekran kapanıyor ancak bu sefer sprite veya herhangi bir grafik göstermemiz mümkün olmuyor. FLD kullanmadan kolay bir şekilde yalnızca karakter ekranını kapatarak DMA'dan kurtulmanın bir yolu var mıdır?

Olmasa ne diye uzun uzun yazayim bu soruyu :) Birbirine benzer iki yolu var bunun. Birincisi 48. raster satırında ($30) $d011'i kapatıp (0'a eşitleyip) 1-2 raster satırı kadar bekledikten sonra yine $d011'in 3 numaralı bitini set edecek olursak (örneğin 8 değerine eşitleyecek olursak) karakter ekranı kapandığı halde spritelar ve bankın sonundaki adres ($3fff, $7fff, $bfff ya da $ffff) görüntülenebilir. Kısacası FLD tekniğinin yarattığı etkinin aynısı yaratılmış olur. Avantajı ise 200 raster satırlık tüm ekranın DMAlarını 1-2 raster satırı zaman harcayarak yok edilebilmesidir.

İkinci yöntem ise bunu ekranın bitiminde 248. raster satırı civarında yapmaktır. Ancak bu durumda ekstra kazancımız alt ve üst borderların da açılmasıdır. Bu alanlara da sprite basabiliriz bu yöntemle.

Şimdi 48.raster satırını kullanarak yaptığım bir örneği inceleyelim. Bu örnekte $3fff adresi kullanılarak iç ekranda dikey çubuklar kaydırılıyor. Bunun haricinde 50. raster satırdan ($32) itibaren 254 raster satırı boyunca hiçbir ekstra zaman tablosu içermeyen bir döngüyle raster çekebiliyoruz. ACME ile derlemenizi öneririm.

```
        !to "nobadlines.prg",cbm

!macro WAIT .v {
        ldy #.v
        dey
        bne *-1
}

        * = $0801
        !byte $0c, $08, $00, $00
        !byte $9e, $32, $30, $36
        !byte $31, $00, $00, $00

        * = $080d
        lda #$01
        sta $3fff
        sei
start:
        lda #$30
        cmp $d012
        bne *-3
        lda #$00
        sta $d011
        +WAIT 24
        lda #$0b
        sta $d011
        lda #$31
        cmp $d012
        bne *-3
        lda #$1b
        sta $d011
        +WAIT 8
        bit $ea
        ldx #$00
loop1:
        txa
        sta $d020
        sta $d021
        +WAIT 9
        inx
        cpx #254
        bne loop1
        asl $3fff
        bne start
        inc $3fff
        jmp start
```

Commodore 64'de sabit renk adresleri nelerdir?

- $0286: Cursor'un o anda sahip olduğu karakter rengi
- $0287: Cursor'un üzerinde bulunduğu arkaplan rengi
- $d020: Border rengi
- $d021: Arkaplan Rengi 0
- $d022: Arkaplan Rengi 1 - Multicolor 0
- $d023: Arkaplan Rengi 2 - Multicolor 1
- $d024: Arkaplan Rengi 3
- $d025: Sprite Multicolor 0
- $d026: Sprite Multicolor 1
- $d027-$d02e: Sprite Renkleri
- $d800-$dbe7: Ekran Renk Belleği

PHP Problemleri

Şekil 2.

Web serverım PHP çalıştırırken arada bir CGI error veriyor ancak refresh edince geçiyor. Problem ne olabilir?

Öncelikle PHP versiyonu eski olabilir. Ancak versiyon güncellendiği halde problem devam ediyorsa, sorun serverınızın çok hızlı olmasıdır. Bu ne tür bir soruna yer açıyor? PHP henüz çıktıyı web servera veremeden web server sayfayı görüntülemeyi deniyor. Bu konuda gidebileceğiniz en güzel çözüm Windows'un "Performans Ayarları"ndan optimizasyonu "Arkaplan Hizmetleri"nden "Uygulamalar"a çevirmektir. Bu seçeneğin yerine gelince;

1. Windows 2000 TR: Denetim Masası -> Sistem -> Gelişmiş -> Performans Ayarları
2. Windows 2000 EN: Control Panel -> System -> Advanced -> Performance Options
3. Windows XP/2003 TR: Denetim Masası -> Sistem -> Gelişmiş -> Performans -> Ayarlar -> Gelişmiş -> İşlemci Zamanlaması
4. Windows XP/2003 EN: Control Panel -> System -> Advanced -> Performance -> Options -> Advanced -> Processor Scheduling

Bu çoğunlukla problemi çözer ancak problemi çözmek için serverın arkaplan hizmetlerine tanınan ve büyük olasılıkla da tanınması gereken önceliğini kaldırarak performansını düşürmüş oluyoruz. Bu da işin negatif tarafı.

Web Server'ıma "localhost" yazarak bağlanamıyorum ancak "127.0.0.1" ile bağlanabiliyorum. Problem ne olabilir?

Sistemin C'ye kurulu olduğunu varsayacak olursak "C:\Windows\system32\drivers\etc" ya da "C:\WINNT\system32\drivers\etc" klasöründe yer alan "hosts" dosyasına bakın. Eğer "127.0.0.1 localhost" şeklinde bir satır göremiyorsanız bu satırı ekleyin.

php.ini'de değişiklik yapıyorum ancak bu değişiklikler algılanmıyor. Sebebi ne olabilir?

1. Eğer PHP'yi CGI modülüyle (php.exe) kurduysanız problem büyük olasılıkla yanlış ini dosyasını editliyor olmanızdan kaynaklanıyordur. Zaman zaman harddisk üzerinde farklı yerlerde php.ini dosyalarının kopyalarıyla karşılaşabilirsiniz. php.exe önce kendi klasörüne, daha sonra windows klasörünün altına bakar ini dosyası için. Normalde php.ini'nin windows klasörünün altında olması tavsiye edilir.
2. PHP dosyalarının içersinde ini_set gibi yaptığınız ayarları değiştiren bir komut kullanılmış olabilir. Bu durumda yaptığınız değişiklikler algılanmayabilir.
3. Bir diğer ihtimal ise PHP'nin ISAPI (php4ts.dll) olarak kurulmuş olmasıdır. CGI ile ISAPI kurulumun arasındaki en temel farklılık CGI'ın her PHP dosyası tetiklendiğinde php.exe'yi yeniden tetiklemesi ve php.exe'nin her defasında php.ini'yi baştan okumasıdır. ISAPI'de ise php4ts.dll web server'a bir plug-in mantığıyla eklenir ve web server çalıştırıldığında bir kereye mahsus php.ini'yi okur ve hafızada kalır. Dolayısıyla php.ini'de yapılan değişikliklerin aktive edilmesi için web server'ı durdurup yeniden başlatmak gerekir.

PHP'de CGI Timeout problemi yaşıyorum. Nasıl çözebilirim?

Bu problemin yine birden fazla kaynağı var. İlk olarak php.ini'de yer alan "max_execution_time" ve "max_input_time" parametrelerini kontrol etmenizi tavsiye ederim. Varsayılan değerleri;

```
max_execution_time = 30
max_input_time = 60
```

şeklindedir. Bunların her ikisini de sıfıra eşitlemek ilk aşamada sorunumuzu çözecektir.

```
max_execution_time = 0
max_input_time = 0
```

Ancak ne var ki bunları sıfıra eşitlediğimiz halde 15 dk gibi bir süre sonunda yeniden CGI Timeout ile karşılaşılabilir. Bu durumda gidilebilecek en kesin çözüm PHP'nin kurulumunu CGI'dan ISAPI'ye çevirmektir.

En son olarak IIS üzerinde de bir "Connection Timeout" ayarı vardır. Varsayılan değeri 900 saniyedir (15 dk). Bu süreyi uzatmak da bir çözüm olabilir. Ancak bu süre uzadığı halde PHP'nin CGI kurulumunda 15 dk'da timeout'a düşebilirsiniz.

emir (at) akaydin (nokta) com

PHP Kursu - 1

Emir 'Skate' Akaydın

Giriş

1997 senesinden beri geliştirilmekte olan PHP dili, 10 yılı aşkın süreçte gün geçtikçe daha popüler ve daha yaygın kullanılan bir dil haline geldi. Özellikle HTML tabanlı web programlamayı hedef alan dilin ASP, CGI, Perl, JSP ve sonrasında ASP.NET dili gibi güçlü rakipleri de oldu (bu rakiplerin bazıları PHP'den daha eskidir). Dilin en büyük avantajlarından biri multi-platform olması yani farklı işletim sistemlerini desteklemesi ve kaynak kodlarının açık gelmesidir.

PHP, büyük ölçüde C, Java ve Perl dillerinin standartlarına uygun olarak geliştirilmiştir. Her ne kadar öncelikli amacı web tabanlı programlamaya yönelikse de, bu dili kullanarak çok daha fazlasını yapmak mümkündür.

Şekil 1.

PHP Dili'nden Nasıl Program Yazılır? Nasıl Çalıştırılır?

PHP dili derleme (compile) gerektirmeyen bir dildir. Bir scripting dili yapısında tasarlanmıştır. Bu dilden bir program yazmak için işletim sistemiyle beraber gelen ya da daha gelişmiş bir text editörü yeterli olacaktır. Yazılan programı ise PHP motoruna parametre olarak verdiğimizde PHP motoru bu programı çalıştıracaktır. Ancak genellikle PHP motoru bu şekilde doğrudan kullanılmaz. Web server'a gerekli tanımlar yapıldıktan sonra web server PHP dilinden yazılmış programın olduğu dosyayı PHP motoruna gönderir ve programın çıktısı web browser ekranından görüntülenir.

Örnekler üzerinden durumu açıklayacak olursak; işletim sistemi Windows, web server IIS, web browser Internet Explorer, text editör ise Notepad olduğunu düşünelim (birçok farklı senaryo kurulabilir).

1. Notepad'i açıp PHP'den programımızı yazıyoruz.
2. ".php" uzantısıyla web serverımızın ulaşabileceği bir klasöre dosyayı kaydediyoruz (IIS için örnek: C:\inetpub\wwwroot\deneme.php).
3. Internet Explorer'ı açarak kendi makine ismimiz/IPmiz ya da lokal isim/IP'yi kullanarak adres çubuğundan dosyaya ulaşıyoruz.
4. Örnek: http://localhost/deneme.php
5. Program çalıştırılıyor ve sonucu browser ekranında görüyoruz.

İlk Program Örnekleri

İlk olarak çok temel seviyede de olsa HTML bilgisine ihtiyaç duyacağız. Bunun için çok basit bir HTML sayfa örneği verelim.

HTML:

```
<html>
<head>
<title>Sayfa Basligi</title>
</head>
<body>
Sayfa #çeri#i
</body>
</html>
```

Bu yukarıda gördüğünüz sayfa örneği HTML formatındaki bir sayfa için minimum iskelet yapısı sayılabilir. Şimdi "Sayfa İçeriği" yazan kısımda alt alta 1'den 3'e kadar olan sayıları yazalım. Unutmamamız gereken nokta HTML'de alt satıra geçmek için "
" komutunu kullanmamız gerekir.

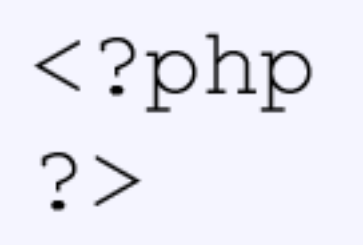
HTML:

```
<html>
<head>
<title>Sayfa Basligi</title>
</head>
<body>
1<br>
2<br>
3<br>
</body>
</html>
```

Bu sayfanın browserdan göreceğimiz çıktısı alt alta 1, 2, 3 rakamlarıdır. Şimdi 1'den 3'e kadar değil de 1'den 10000'e kadar olan sayıları yazalım. Bu durumda üç yöntemimiz var.

1. Bu iş için adam tutup para vereceğiz kendisine, tek tek yazacak 1'den 10000'e kadar satırları.
2. Herhangi bir programlama dilinden istediğimiz HTML çıktısını veren bir program yazacağız.
3. PHP, JavaScript, VBScript gibi bir scripting diline başvurup işi HTML'in içine bir script bloğu açarak halledeceğiz.

Evet bu işi yapmamızın tek yolu PHP değil. Javascript, VBScript gibi dillerle de bu işi halledebiliriz. Ancak bu diller ile PHP'nin farkına birazdan değineceğiz. Şimdi PHP ile bu işin nasıl yapıldığını görelim.



Şekil 2.

PHP:

```
<html>
<head>
<title>Sayfa Basligi</title>
</head>
<body>
<?php
for($i=1;$i<=10000;$i++)
    echo $i."<br>";
?>
</body>
</html>
```

Böylelikle klasikleşmiş "Merhaba Dünya" örneğinden önce farklı bir örnek ile yola çıkmış olduk. Bu örnekte şimdilik dikkat etmeniz gereken nokta sayfa içeriğinin yer aldığı bölgeye nasıl bi program bloğu açtığımız konusu. PHP dilinde en yaygın ve doğru kullanım "<?php" ve "?>" tagleridir. Aslında PHP'nin ayarlarında "short tag" kullanımı açık ise "php" kelimesi yazılmadan "<?" ve "?>" şeklinde blok açılıp kapanabilir, hatta başka alternatifler de mevcuttur. Yine de en güvenli ve her durumda çalışacak yöntem "<?php" şeklinde bloğu açıp "?>" ile kapatmaktır. Arada kalan bölge artık HTML değil PHP kodlarımızın yer aldığı bölgedir.

Yukarıda verdiğimiz örnek 1'den 10000'e kadar "i" değişkenini saydıran ve her defasında i'nin değeri ve "
" HTML komutunun çıktısını veren bir örnektir.

PHP ile JavaScript/VBScript'in Farkı

Temelde PHP de Javascript ve VBScript de birer scripting dilidir.

Scripting kavramını kısaca açıklamak gerekirse, bir programa çekrideğine eklenti olan ve projeyi yeniden derlemeden kolayca değiştirilebilen program parçalarına denir. Örneğin bazı bilgisayar oyunları son kullanıcıya ulaştıktan sonra yeni bölümler programın exesi değiştirilmeden yalnızca scriptler eklenerek oyuna ilave edilebilir. Hatta son kullanıcıya şahsen müdahele hakkı bile tanınabilir. Benzer biçimde birçok IRC programı (örneğin MIRC), kullanıcıya chat serverlarında ve chat odalarında yapmak istedikleri şeyleri programlayabilmeleri için scripting desteğiyle gelirler.

Az önce verdiğimiz 1'den 10000'e kadar sayı saydırıp ekrana yazdırma örneğinin PHP haricinde JavaScript ve VBScript ile de yapılabileceğinden bahsetmiştik. Örnekleri aşağıda yer almaktadır.

JavaScript:

```
<html>
<head>
<title>Sayfa Basligi</title>
</head>
<body>
<script type="text/javascript">
for(i=1;i<=10000;i++)
    document.write(i+"<br>");
</script>
</body>
</html>
```

VBScript:

```
<html>
<head>
<title>Sayfa Basligi</title>
</head>
<body>
<script type="text/vbscript">
For i = 1 To 10000
    document.write(i & "<br>")
Next
</script>
</body>
</html>
```

Bu iki script PHP'den yapmış olduğumuz örnekle aynı sonucu verirler. Ancak çalışma yapıları arasında PHP'ye kıyasla çok büyük farklılıklar yer almaktadır. Çünkü JavaScript ve VBScript bu şekilde kullanıldıklarında istemci tarafında (client-side) çalışırlar. Kısacası web server, web browser'a bir çıktı verdiğinde bu scriptler de çıktının içinde olduğu gibi yer alır ve browser sayfayı göstermek istediği anda çalıştırılarak sonuç üretirler. PHP'de ise herşey sunucu tarafındadır (server-side). PHP scripti sunucuda çalıştırılır ve çıktısı web browsera gelir.

Şekil 3.

Bunlardan hangisi daha iyi sonuç verir?

PHP oldukça hızlı bir dildir. Dolayısıyla her tür kıyaslamada JavaScript/VBScript'e göre daha iyi sonuç verecektir. Ayrıca PHP yalnızca text output vermekle ilgilenir. İstemci tarafında çalışan diğer scripting dilleri browsera bağımlı başka birçok nesneyle de ilgilenmek zorundadırlar. Fakat burada gelmeye çalıştığımız asıl nokta bu değil. Test ortamında kendi makineniz üzerinde çalışıyor olabilirsiniz. Ancak yaptığınız PHP programı/web sitesi yayına girdiği andan itibaren sunucu ve istemci aynı makine olmaktan çıkacaktır. Bu durumda istemcilerin makinelerinin değişken performanslarının sitenizin çalışma hızına etkisini minimuma indirmek için yukardaki gibi örneklerde PHP kullanmak çok daha mantıklı bir yöntem olacaktır. Zamanla PHP kullanmamızı mecburi kılacak veritabanıyla iletişim gibi konulara da geleceğiz. Ancak PHP kullanmamız zorunlu olmayan birçok durumda dahi PHP doğru seçim olacaktır.

Ne zaman hangi dili kullanmamız lazım?

Bu sorunun cevabını verebilmek için PHP'nin bize tam olarak ne sağladığını anlamak lazım. PHP, kendisine ait scriptleri sunucu tarafında çalıştırıp, bunlardan oluşturduğu sonuçlarla birlikte bir HTML çıktı üreterek istemciye yollar. Yani bir PHP sayfası çalıştıktan sonra browser'da "kaynağı görüntüle" (view source) seçeneği seçildiği zaman PHP kodlarını değil, bu kodların ürettiği sonuçlardan oluşmuş HTML kodlarını görürüz.

Bu iki anlama gelir.

1. PHP dili kaynak kodları istemciden bağlanan kişilerden korumak açısından güvenlidir. Sunucuya erişim yetkisi olmayan kişiler tarafından görüntülenmesi imkansızdır. (Hatta Zend firmasının uygulamaları ile sunucuya erişim yetkisi olan kişilerden de gizlenebilir). İstemci tarafında çalışan scripting dillerinin kodları ise açıktır, standart yöntemlerle gizlenemez.

2. PHP bütün işi sunucu tarafında yapar ve istemciye ulaştıktan sonra PHP'nin artık hiçbir fonksiyonu kalmamıştır. Bu yüzden PHP ile dinamik bir sayfa oluşturmak, browserda olan olayları PHP'ye yeni bir sayfa yükletmeden göndermek, sonuçlarını almak mümkün değildir. AJAX ve benzeri yöntemler ile sayfa güncellenmeden PHP'den veri çekilebilir ancak yine istemci tarafında çalışan dillerin yardımına ihtiyaç vardır. Kısacası bu tür durumlarda JavaScript, VBScript gibi dillerin faydasını görmekteyiz. Bu diller browserla ve HTML nesneleriyle doğrudan etkileşim içersindedirler. Yeni bir sayfa yükletmeden o sayfa içersindeki herşeyi kontrol edebilir, kullanıcı ile interaktif bir şekilde çalışabilirler.

Yani amacınız bir linke ya da butona tıklanınca o sayfa içersinde bir mesaj çıkartmak, bir resmin üzerine mouse ile gidince resmin değişmesi gibi konular ise PHP yanlış bir seçimdir. Ancak sayfa oluşturulurken veritabanından okunan değerlerin ekrana yazdırılması, HTML formlarından gönderilen verilerin veritabanına kaydedilmesi ya da sayfanın sabit kısımlarına (örneğin menü) her dosyada yer vermek ve güncellerken tüm dosyaları güncellemek yerine tek bir dosyadan yükletmek gibi amaçlarda PHP doğru seçim olacaktır.

Merhaba Dünya Örneği

Şekil 4.

Birçok dilde ilk öğrenilen şey ekrana nasıl yazı yazdırıldığıdır. Elbette ki PHP'de de bu çok önemli bir konudur ve şimdi bu konuyla ilgili detayları göreceğiz.

PHP'de bunu yapmanın birden fazla yolu mevcuttur. İlk olarak en temel yazı yazdırma örneğini görelim.

echo

```
<html>
<head>
<title>Sayfa Basligi</title>
</head>
<body>
<?php
echo "Merhaba Dünya";
?>
</body>
</html>
```

"echo" komutu yazı yazdırmak için PHP'deki en genel ve en hızlı yöntemdir. Bu komut doğrudan PHP'nin çekirdeğinde yer alan bir komuttur. Hiçbir değer döndürmez. Bu fonksiyonu çağırırken parantez kullanmak gerekmez. Parantez kullanılmadığı durumda virgüllerle ayrılarak birden fazla parametre alabilir (bu parametrelerin değerlerini yan yana yazar).

Diğer alternatifler ise Basic'den alışık olduğumuz "print" ve C'den alışık olduğumuz "printf" komutlarıdır. Bunlardan "print" "echo" ile büyük benzerlikler taşır. Hız olarak hemen hemen "echo" ile aynıdır. "print"in fazladan bir sonuç döndürdüğü düşünülecek olursa tesbit edilmesi güç derecede "echo"dan daha yavaştır diyebiliriz. Ayrıca "echo" gibi birden fazla parametre alma özelliğine sahip değildir.

print

```
<html>
<head>
<title>Sayfa Basligi</title>
</head>
<body>
<?php
print "Merhaba Dünya";
?>
</body>
</html>
```

Son olarak "printf" komutunu ele alacak olursak "printf", "echo" ve "print"den epey farklı bir komuttur. C kütüphanesindeki "printf" ile hemen hemen aynı şekilde çalışır. PHP'den ilerki konularda göreceğimiz şekilde değişkenler belirli bir tipe bağımlı olarak tanımlanmazlar ve dinamik olarak tip değiştirebilirler. Ancak printf komutu ile C'deki gibi değişkenleri formatlayabiliriz. "Merhaba Dünya" örneğine geri dönecek olursak printf de diğerlerinden çok farklı bir kullanıma sahip değildir.

printf

```
<html>
<head>
<title>Sayfa Basligi</title>
</head>
<body>
<?php
printf("Merhaba Dünya");
?>
</body>
</html>
```

Bunların haricinde ekrana çıktı veren vprintf(), exit() gibi başka komutlar da yer almaktadır. Ancak "Merhaba Dünya" örneği için şimdilik bu kadarı yeterli.

Değişkenler

PHP'de değişken kullanımı diğer birçok dile göre çok daha basittir. Ancak PHP'nin değişkenleri nasıl yorumladığı iyi anlaşılmadığı zaman bazı durumlarda programcı problemlerle karşılaşabilir. Bu sebeple PHP'de yer alan değişken tipleri, değişkenlerin yapısı, değişken tip dönüşümleri bilinmesi gereken önemli konulardır.

PHP'de desteklenen değişken tipleri şunlardır:

1. Boolean: Doğru/Yanlış
2. Integer: Tam Sayı
3. Float: Ondalıklı Sayı
4. String: Metin
5. Array: Dizi
6. Object: Nesne
7. Resource: Kaynak
8. NULL: Boş

PHP'de bu değişken tipleri olduğu halde genellikle "değişken tipi" ile ilgili çok fazla bir işlem yapılmaz. Çünkü PHP'de değişken tanımlama zorunluluğu yoktur. Bir değişken, eşitlendiği değere göre otomatik olarak tip belirleyebilir ve daha sonra yine otomatik olarak tip dönüşümü yapabilir. Bu özellik PHP'yi programcılar için çok daha kolay kullanımlı bir dil haline getirmiştir. Zamanla çeşitli örneklerle bunları göreceğiz.

Şimdi PHP değişkenlerinin bazı özelliklerini tanıyalım.

PHP değişkenleri başlarında dolar işaretiyle ($) kullanılırlar.

Büyük küçük harf hassasiyetine sahiplerdir, yani $a ve $A iki farklı değişken olarak algılanır.

Değişken isimleri a'dan z'ye bir harf ya da alt çizgi (_) karakteri ile başlayabilir. Rakam ya da başka bir sembol ile başlayamaz.

$this özel bir değişkendir. Bir değere eşitlenemez.

Değişkenlerin başında kullanılan $ sembolünün yardımıyla PHP'de değişkenin değerinden başka bir değişken oluşturabilme özelliği vardır. Buna PHP'de "değişken değişkenler" (variable variables) ismi konulmuştur. $ sembolleri iç içe istenilen sayıda kullanılabilmektedir. Örnek:

değişken değişkenler örnek 1

```
<?php
$a = "abc";
$$a = "123";
echo $abc;
?>
```

Bu program ekrana "123" yazacaktır. Dikkat ederseniz ekrana yazdırılan değişken $abc olduğu halde programın içinde "abc" bir değişken olarak tanımlanmamış durumda. Yani $abc = "123"; şeklinde bir atama yapılmamış. Fakat $a değişkenine string olarak "abc" değeri yüklendikten sonra $$a "123"'e eşitlenmiş. $$a'yı $($a) şeklinde düşünecek olursak $a'nın değeri "abc" olduğu için $$a = $abc anlamına gelmektedir. Bu daha da iç içe ilerletilebilir. Örnek:

değişken değişkenler örnek 2

```
<?php
$a = "abc";
$$a = "def";
$$$a = "ghi";
$$$$a = "123";
echo $ghi;
?>
```

Satır sırasıyla $a "abc"'ye, $abc "def"'ye, $def "ghi"ye ve $ghi ise "123"e eşitlenmiş oldu. Burada $$$$a yerine $$$abc, $$def ya da $ghi kullanabilirdik. Tek bir $ kalana kadar sağdan sola doğru değişkenlerin değerlerini yerine koyarak ilerlediğimizde bu sonuca varıyoruz.

PHP'de otomatik olarak yapılan değişken dönüşümlerine bazı örnekler verelim.

değişken dönüşümleri

```
<?php
$a = "15"; // $a string tipi bir degisken
           // olarak tanimlandi ve degeri
           // "15" oldu

$a += 5;   // $a'nin tipi integer'a
           // donustu ve degeri 20 oldu

$a -= 1.75;  // $a'nin tipi float'a donustu
             // ve degeri 18.25 oldu

$b = 100 + "55"; // $b degiskeni integer
                 // olarak tanimlandi ve
                 // degeri 155 oldu

$c = 100 + "55 abc"; // $c degi#keni integer
                     // olarak tanimlandi ve
                     // degeri 155 oldu

$d = 100 + "abc"; // $d degi#keni integer
                  // olarak tanimlandi ve
                  // degeri 100 oldu
?>
```

İlk aşamada değişkenlerle ilgili bilmemiz gereken özellikler bunlardır. Konular ilerledikçe değişkenlerin farklı özelliklerini de göreceğiz.

Fonksiyonlar

PHP'de fonksiyon kullanımı diğer dillerle kıyaslandığında oldukça standarttır. Ancak PHP'nin yapısında gelen bazı özel fonksiyonlar sayesinde fonksiyon kullanımı çok güzel bir boyut kazanmıştır. Şimdi basit bir fonksiyon örneğiyle başlayalım.

topla() fonksiyonu

```
<?php
$sayi1 = 3;
$sayi2 = 5;
$toplam = topla($sayi1, $sayi2);
echo $toplam;

function topla($a, $b)
{
    return $a+$b;
}
?>
```

Bu programın sonucunda ekrana 8 rakamı yazdırılır. Burada programı ana program ve fonksiyon şeklinde iki parçada incelemek lazım.

Ana program:

1. 3 değeri $sayi1'e atanır.
2. 5 değeri $sayi2'ye atanır.
3. topla fonksiyonundan dönen sonuç $toplam'a atanır.
4. $toplam ekrana yazdırılır.

Fonksiyon:

1. Ana programda verilen $sayi1 ve $sayi2 değişkenlerinin değerleri $a ve $b parametrelerine atanır.
2. $a ve $b'nin değerleri toplanarak sonuç ana programa geri döndürülür.

Burada dikkat edilmesi gereken noktalardan biri PHP'de fonksiyonların önceden tanımlanmadan kullanılabildiğidir. C/C++ gibi dillerde kaynak kodlar içersinde yukardan aşağı doğru bir sıralama olması gerekmektedir. Yani "topla()" isimli bir fonksiyon kullanılacak ise, bu fonksiyonun kullanıldığı yerden önce tanımlanmış olmalıdır. Bunun iki yöntemi vardır. Birincisi fonksiyonu kullanıldığı yerden daha önceye yazmak. İkincisi ise fonksiyon prototipi dediğimiz fonksiyonun yalnızca tanımının bulunduğu bölümü yine kullanıldığı yerden önceye, tercihen programın en başlarında bir bölgeye yazmak (bu örnekte "function topla($a, $b)" prototip kabul edilebilir). Fakat daha önce de belirttiğimiz gibi PHP böyle birşeye ihtiyaç duymaz. Fonksiyonu kodun istediğiniz bölümüne yazabilir ve istediğiniz yerden çağırabilirsiniz.

Fonksiyonların içersinde kullanılan değişkenler lokal değişkenlerdir. Yani ana programda tanımladığımız değişkenlerin değerleri fonksiyon içersinde geçersizdir ve aynı değişken ismi fonksiyonun içersinde kullanılsa da ana programdaki değişkenlerin değerlerini bozmazlar. Örnek:

lokal değişken örneği 1

```
<?php
$sayi1 = 3;
$sayi2 = 5;
$toplam = topla();
echo $toplam;

function topla()
{
    return $sayi1+$sayi2;
}
?>
```

Sonuç olarak ekrana hiçbirşey yazdırılmayacaktır. Çünkü $sayi1 ve $sayi2 fonksiyon dışında tanımlanmıştır. Şimdi başka bir örnek ile lokal değişken kullanımını iyice pekiştirelim. Bu örnekte HTML kodlarını da örneğe dahil edelim.

lokal değişken örneği 2

```
<html>
<head>
<title>Lokal De#i#ken Örne#i</title>
</head>
<body>
<?php
$a = 100;
echo "Ana programda a'nin degeri: ".$a."<br>";
test();
echo "Ana programda a'nin degeri: ".$a."<br>";

function test()
{
    $a = 25;
    echo "test() fonksiyonunda a'nin degeri: ".$a.
         "<br>";
}
?>
</body>
</html>
```

Bu programın çıktısı şu şekilde olacaktır.

```
Ana programda a'nin degeri: 100
test() fonksiyonunda a'nin degeri: 25
Ana programda a'nin degeri: 100
```

Sırasıyla PHP kod satırlarını inceleyecek olursak;

1. $a değişkeni 100 değerini alıyor.
2. $a'nın değeri 100 olarak ekrana yazdırılıyor.
3. test() fonksiyonu çağırılıyor.
4. Fonksiyon içersinde $a değişkeni 25'e eşitleniyor.
5. $a'nın değeri 25 olarak ekrana yazdırılıyor.
6. Fonksiyondan ana programa dönüş yapılıyor ve kaldığı yerden program çalışmaya devam ediyor.
7. $a'nın değeri birkez daha ekrana yazdırılıyor.

İşte burada kritik nokta 7. adım. Bu adımda $a'nın değeri 100 mü olmalı 25 mi? Sonuç belli, 100 oluyor. Peki neden? Çünkü fonksiyon içersinde yer alan $a ana programdaki $a ile adaş olduğu halde iki farklı değişken olarak işlem görürler.

Şimdi ana programda tanımlanmış değerleri ilk fonksiyon örneğinde yapmış olduğumuz gibi parametre kullanmadan fonksiyonlar içersinde kullanmanın yolunu göreceğiz. Bunu "global" komutu ile yapıyoruz. Fonksiyon içersinde global olarak kullanılmak istenen değişkenler başına "global" yazılarak tanımlanıyorlar. Daha sonra bu değişkenler ana programdaki değişkenle aynı değişken olarak kullanılabiliyorlar. Son verdiğimiz örneği bir de bu şekilde inceleyelim.

global komutunun kullanım örneği 1

```
<html>
<head>
<title>Global Komutu Kullan#m Örne#i 1</title>
</head>
<body>
<?php
$a = 100;
echo "Ana programda a'nin degeri: ".$a."<br>";
test();
echo "Ana programda a'nin degeri: ".$a."<br>";


function test()
{
    global $a;
    $a = 25;
    echo "test() fonksiyonunda a'nin degeri: ".$a.
         "<br>";
}
?>
</body>
</html>
```

Bu örnekte çıktı şu şekilde olacaktır.

```
Ana programda a'nin degeri: 100
test() fonksiyonunda a'nin degeri: 25
Ana programda a'nin degeri: 25
```

Aslında bir önceki örnekten tek farkı "global $a" satırı. Ancak sonuç olarak fonksiyon çağrıldıktan sonra ana programda ekrana yazılan değer fonksiyonun içersinde atanan değere dönüştü. Sebebi ise bu defa işlerin şu şekilde yürümüş olması;

1. $a değişkeni 100 değerini alıyor.
2. $a'nın değeri 100 olarak ekrana yazdırılıyor.
3. test() fonksiyonu çağırılıyor.
4. $a değişkeni global olarak tanımlanıyor. Artık $a lokal değil global bir değişken.
5. Global olarak tanımladığımız $a değişkeni 25'e eşitleniyor.
6. $a'nın değeri 25 olarak ekrana yazdırılıyor.
7. Fonksiyondan ana programa dönüş yapılıyor ve kaldığı yerden program çalışmaya devam ediyor.
8. $a'nın değeri birkez daha ekrana yazdırılıyor.

Bu defa araya fazladan bir adım girmiş oldu. 8. adıma geldiğimizde ise $a'nın değeri fonksiyon içersinde atanan değere eşit oldu.

Peki "global" komutu bize ne sağlar? Bir fonksiyona parametre olarak verilmeyen ancak program içersinde global olarak anlam ifade eden değişkenleri fonksiyon içersinden kullanmamızı sağlar. Henüz veritabanı konusuna gelmemiş olsak da buna verilebilecek en güzel örneklerden biri budur. Veritabanı bağlantısı yaptığımızda bize bağlantı bilgilerinin tutulduğu bir değişken döner (genellikle resource tipinde). Bu değişkeni veritabanı üzerinde işlem yapan tüm fonksiyonlara bir parametre olarak vermek yerine fonksiyonların içersinde global komutuyla kullanmak çok daha pratiktir.

Şimdi ilk fonksiyon örneğimiz olan "topla()" örneğinin parametresiz olarak, global komutuyla nasıl kullanıldığını görelim.

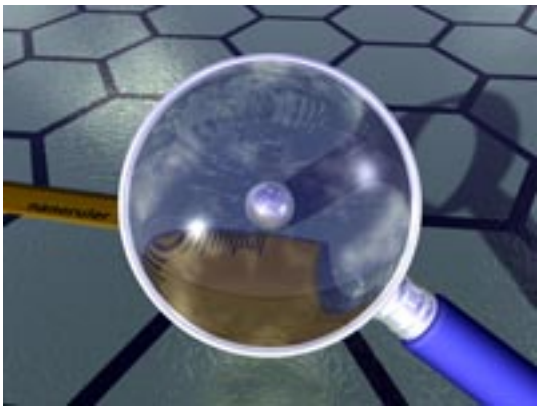
global komutunun kullanım örneği 2

```
<html>
<head>
<title>Global Komutu Kullanim Ornegi 2</title>
</head>
<body>
<?php
$sayi1 = 3;
$sayi2 = 5;
$toplam = topla();
echo $toplam;

function topla()
{
    global $sayi1;
    global $sayi2;
    return $sayi1+$sayi2;
}
?>
</body>
</html>
```

Bu programın çıktısı da 8 olacaktır. Dikkat ettiyseniz "topla()" fonksiyonu parametre almıyor ve $sayi1 ve $sayi2'nin değerlerini fonksiyon dışında tanımlanan global değişken değerlerinden alıyor.

PHP'de fonksiyonlar değer döndüren ve değer döndürmeyen şeklinde ikiye ayrılmazlar. Yani tanımlanışları açısından bir fark bulunmamaktadır. Eğer siz bir fonksiyonda "return" komutunu kullanır ve yanında bir değer ya da değişken yazarsanız bu fonksiyon bir değer döndürecektir. "return" komutu bulunmayan fonksiyonlar ise bir değer döndürmezler, sadece fonksiyon bloğunda yer alan programı çalıştırıp çağırıldıkları yere geri dönerler.

Bu Bölümün Değerlendirmesi



Şekil 5.

Bu ilk bölümde PHP dilinin genel yapısına bir göz attık, değişkenler ve fonksiyonlar konusuna giriş yaptık. Önümüzdeki bölümlerde PHP'nin daha güzel özelliklerine göz atacağız ve yavaştan PHP'yi işe yarar biçimde kullanmaya başlayacağız. Şimdilik editörüm Nightlord'a bir iyilik yaparak yazıyı daha uzatmıyor ve burada noktalıyorum. Gelecek sayıda görüşmek üzere.

emir (at) akaydin (nokta) com

C64 Piksel Grafik Kursu - 2

Kürşad 'Hydrogen' Karamahmutoğlu

Çok renk çizim teknikleri

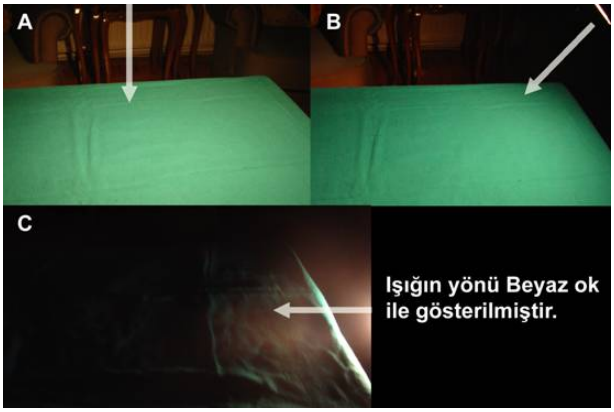
Plazma'nın ilk sayısında çok renk çizim tekniklerine giriş yapmıştık. Bu sayıda C64'de pikselleme dersine biraz ara verecek ve iyi dijital resim çizebilmemiz için gerekli olan bazı temel bilgilerin üzerinden geçmeye başlayacağız. Hepimizin bildiği gibi görüntüyü oluşturan kavram ışık olduğu için, işe ışığın karakterini incelemekle başlamamız faydalı olacaktır.

Işığın cisimler üzerindeki etkileri

Aydınlatma etkisi

Gerçek yaşamda masamızın üzerinde duran düz, yeşil bir örtüyü ele alalım. Bu örtü pencereden giren ışıkla aydınlansın. Bildiğimiz gibi üzerine ışık düşmeyen cisim siyah görünür. Üzerine düşen ışık miktarı arttıkça, cisimlerin renk özellikleri ön plana çıkmaya başlar. Ve çok kuvvetli ışıklara doğru gidildikçe bu renk beyaza yakınlaşır. Cisimlere vuran beyaz ışık, bir cismin gerçek rengini ortaya çıkarır. Gelen ışığın rengine göre, materyalin rengi de değişim gösterir. Masa örtümüz düzgün bir şekilde masa yüzeyine yayılmış olsaydı ve ışık kaynağımız masa yüzeyine oldukça uzak bir yerden homojen ve yüzeye dik olarak gelseydi (gün ışığı gibi) o takdirde, biz masamızın düz yeşil rengini örtünün her bölgesinde eş şiddette görebilecektik.Eğer aynı ışık kaynağı, dik olarak değil de mesela 45 derece açıyla vursaydı masaya, bu takdirde yüzeye vuran toplam ışık miktarı azalacağından dolayı örtü biraz daha koyu yeşil bir renkte görünecekti.

Eğer ışık kaynağımız masayla paralel yerleşseydi, o zaman yüzey hiç ışık alamayacağından dolayı örtü siyah görünecekti.



Şekil 1. Işığın yönüne göre aydınlanma

Sizlere kendi çektiğim fotoğraflardan örnek verebilmek için, yazıdaki homojen yayılımlı beyaz güneş ışığını,homojen olmayan sarımtırak masa lambası ile değiştirmek durumunda kaldım. Masa lambasının aydınlatma şiddeti yönlendirildiği noktalardan kenarlara doğru azalsa da, bu örneğimiz için çok büyük bir sorun teşkil etmeyecek.

A fotoğrafında yüzeye dik vuran sarı masa lambası ışığı, B fotoğrafında yüzeye 45 derece açı yapacak şekilde çevrilmiş. A fotoğrafındaki aydınlanma miktarının B'ye göre daha fazla olduğuna ve ışığın renginin A fotoğrafında daha belirgin olduğuna dikkat çekiyoruz. C fotoğrafında ise, ışık masaya paralel yerleştirilmiş. Işık, yüzeyle kesişmediği için örtü hemen hemen hiç ışık almıyor. (çevreden yansıyan bir miktar ışık, masa yüzeyini az bir miktar aydınlatıyor)

Şimdi bu örtüyü biraz kırıştırarak masamızın üzerine serelim. Işık kaynağımızı yine çok uzaktan masaya dik olarak vuracak şekilde ayarlayalım. Üzerindeki kırışıklıklardan dolayı örtümüzün bir çok bölgesi farklı açılardan ışık alacak. Ve farklı şiddetlerde aydınlanacak. Bu da örtü üzerinde yeşilin, açıktan koyuya bir çok tonunun oluşmasına neden olacak. Işığa dik konumlanan yüzeylerde aydınlanma etkisi maksimuma ulaşacak, yüzey ile yapılan açı azalmaya başladığında ise, aydınlanma etkisi de azalmaya başlayacak. Işığın doğrultusunun yüzeye paralel olduğu yerler ise, yüzey hiç ışık almayacağından ötürü, karanlıkta kalacak.

Şekil 2. Kırışıklıkların aydınlanışı

Şekil 3. Kırışıklıkların aydınlanışı

Yansıma etkisi

Eğer ışığın gerçek yaşamdaki etkisi sadece aydınlatma olsaydı, cisimlerin ışık görmeyen yüzeyleri hiç aydınlanmazdı.. Ve gerçek yaşamda bol bol sıfır karanlıkta, yüksek kontrastlı yüzeyler görürdük. Ancak ışık aynı zamanda vurduğu yüzeyden yansıma özelliğine de sahiptir. Işık ışınları bir yüzeyden yansırken, materyalin yansıtma özelliğine bağlı olarak, yansıdığı yüzeyin bir miktar rengini de alır. Masa üzerine yayılmış, kırışmış yeşil örtümüzün direk ışık almayan yüzeyleri, ışık alan yüzeylerden yansıyan ışıklarla, daha az miktarda da olsa aydınlanır. Ve ışık almayan yüzeye beyaz bir kağıt parçası koyarsak, bu kağıt parçası, ışınlar masa örtüsünden yansıdığı için çok çok az bir miktar yeşil görünecektir. Yansıtma özelliği çok yüksek olan materyaller ayna özelliği göstereceklerdir.

Kırılma etkisi

Işık, maddelerin geçirgenlik özelliklerine bağlı olarak ortam yoğunluk değişimlerinde kırılır. Az yoğun havadan, çok yoğun fakat saydam su ortamına geçildiğinde bu etki bariz olarak farkedilebilir. Işığın bu kırılma olayı, nesnelerin olduğundan farklı görünmelerine neden olur. Görüntüyü oluşturan ışık olduğuna göre,ışığın normalden farklı davranması görüntünün gerçekte olduğundan farklı olmasına neden olur.

Gölge etkisi

Çok tabii bir olay olarak, ışık ile masa örtümüz arasında, başka bir cisim var ise, bu cisim masa örtümüzün alacağı ışığı engelleyecek, kendi şeklinin bir iz düşümünü masa örtümüzün üzerinde bırakacaktır. Cisim ışık kaynağına ne kadar yakın ve masadan ne kadar uzak ise bu izdüşüm o kadar büyük olacaktır. Bu olaya gölge etkisi denir. Ortama gelen ışık kaynağının hacimsel büyüklüğü ve-veya çevreden yansıyan ışık ne kadar fazla ise, gölgelerin sert hatları da o oranda yumuşaklaşır (Yumuşak gölgeler). Gerçek yaşamda ışık çok sayıda yüzeyden yansıdığı için yumuşak gölgeler daha ağırlıktadır. Sert gölgeler daha ziyade gece, küçük tekil ışık kaynağı ile aydınlatılan sahnelerde belirginleşir. Kasvet yaratılmak istenildiğinde de, sert gölgeler tercih edilebilir. Aynı şekilde kırışmış masa örtümüzün kırışık yüzeyleri de, kendi üzerinde pek çok gölgeye sebebiyet verecektir.

Materyallerin ışığa verdikleri tepkiler

Doğada bulunan bir çok materyal türü değişik aydınlanma, yansıtma ve kırılma (saydam nesnelerde) karakterleri gösterirler. Ayrıca yukarıda açıklamadığımız ışık geçirgenliği yani opaklık da önemli bir materyal özelliğidir. Bütün bu özellikler bir materyalin görsel karakterini belirtir. Örneğin metal bir kılıç, ışığın önemli bir bölümünü yansıttığından parlak görünürken, bir ağaç kabuğu, daha az yansıtma özelliğine sahiptir ve oldukça mat görünür. İşte bir materyal için en belirgin özellik, rengi ve şeklinden çok, ışığın özelliklerine verdiği tepkidir. Her ikisi de aynı renk ve desende olan düz ipek bir kumaş ile düz kadife bir kumaşı ayırmamıza olanak veren şey, ışığa karşı verdikleri reaksiyondur. Özellikle yüzeye belli bir açı ile vuran ışık, bu etkilerin daha da belirginleşmesini sağlar. Bu etkilerin bir resimde doğru bir şekilde taklit edilmesi resmin gerçekçi ve inandırıcı olmasını sağlar. Ancak insanlar, fizik problemi çözer gibi resim yapamayacaklarından, bu etkileri sezgisel yöntemlerle, gerçeğe (realist) veya duygulara (sürrealist) yakın olarak resmetmeye çalışmak, genelde ressamların yaptığı iştir. Bizim amacımız da teorik bilgi vermekten çok, piksel çizim üzerine püf noktalarına değinmek, dolayısıyla bir an önce bu etkilerin piksellerle nasıl anlatılacağına yönelsek iyi olacak.

Işığın materyal üzerindeki etkilerinin, resmin etkisini arttıracak şekilde taklit edilmesi

Ortamın rengi: Bir ortamdaki cisimlerin çoğu kendisine yakın olsun olmasın bütün ışık kaynaklarından ve materyallerden etkilenirler. Ortama giren ışıkların, ortamdaki materyallerin renklerinin toplamı ortamı dolduran hakim rengi ortaya çıkarır. C64'de

çizim yaparken, ilk bölümlerde belirttiğimiz gibi kısıtlı renk kullanımına sahip olduğumuzdan, hakim rengin seçimi (veya bilinmesi) bizler için ekstra önemlidir. Çünkü hakim renk sizin en çok kullanacağınız renklerden biridir.

Püf noktası: İlk bölümde multi-colour formatında, 4x8'lik hücrelerde, biri bütün satırlar için ortak ve diğer ikisi her satırda değişebilen 3 renk kullanabileceğimizi belirtmiştik. Eğer bütün satırlarda ortak olan rengi, resimdeki hakim renge denk düşecek şekilde tasarlarsak, çok büyük bir avantaj elde etmiş oluruz.

Ayrıca, renk değişimi en çok geçiş tonlarında sorun çıkardığından, ortak rengi, çok kullanılan bir geçiş rengi olarak tanımlamak da, büyük bir avantajdır. Maalesef, utility programcıları çizer hayatını bu kadar kolaylaştıran triklerden bihaber oldukları için, bir çok programda, 4x8'lik hücre içindeki renklere sonradan müdahale yapılamaz.

Eğer bir resmin bir bölgesini kırmızı bir ışık aydınlatıyorsa, sahneyi tamamlamak için benzer bir kırmızı tonunu insan gözü çeşitli yerlerde arar. Normal koşullarda resmin bir bölgesindeki hakim rengin, başka yerleri etkilememesi seyrek gerçekleşen bir olaydır. Kırmızı ile aydınlanmış bölgeye çok yakın bölgelerin nasıl bir aydınlanmaya maruz kaldığını insan gözü tespit edebilir ve resmin o bölgesini boyayarak gerçekçi görüntü oluşturabilir. Ancak resmin hesaplanamayacak bölgelerine bile bir miktar kırmızı koymak, hem daha beklenilen, hem de daha şık ve estetik bir görüntü ortaya çıkaracaktır. Bu işi biraz da realizmden bağımsız olarak "Kahverengi ayakkabı ve kahverengi çantanın uyumu" gibi düşünebilirsiniz. Bazen estetik olanı boyamak, doğru olanı boyamaktan daha etkilidir.

Şimdi bu konuya, gayet mükemmel piksellenmiş, Cyclone'un Deus Ex Machina isimli ünlü Crest demosundaki resmini inceleyerek bakalım.

Şekil 4. Deus Ex Machina / Cyclone

Bu harika I-Fli piksel çizimde, gene de gözümüze eksik gelen bir şey var. Ekranı sol ve sağ olarak ikiye bölelim. Önce sağ tarafı, sonra sol tarafı bir karton yardımıyla kapatalım. Sol taraf kendi içerisinde oldukça tutarlı görünüyor. Sağ taraf da öyle. Solda soğuk renkler ağırlıktayken sağda sıcak renkler ön plana çıkmış. Ancak bu ikisi çok inandırıcı şekilde birleşmemiş. Sağda astronotun bulunduğu ortam betimlenmiş ancak o bölgedeki sıcak renklerin, yansıtıcı özellikleri oldukça yüksek metalik yüzeyleri, camı ve saçları pek fazla etkilememiş olması bir kopukluk yaratıyor.

Bu sayıda, biraz kısıtlı bir zamanda bu yazıyı hazırlamaya çalıştığım için biraz ani bir bitiş yapmak zorunda kalıyorum. Gelecek sayıda, piksellere biraz daha fazla gömüleceğiz ve daha heyecan verici şeylerden bahsedeceğiz. Plazma #5'de görüşmek üzere.

Düzeltme: Plazma #1'de yayınlanan ilk bölümde yaptığım bir hatayı düzeltmek istiyorum. Coşkun bir ruh hali ile yazıları yazan yazarınız Hydrogen burada kopmuş gitmiş biraz. Kimse de uyarmamış onu. Neyse ki tekrar okuyunca fark ettim ve fazla vakit geçmeden düzeltiyorum."Hires-Interlace (HI)Çizim Modu: 320x200 çözünürlüğünde iki hi-res grafik ekranının ard arda gösterilmesiyle oluşturulur. Yeni çözünürlük 640x200'e çıkar. 16x8'lik piksellik hücrede mantıklı olarak biri backround, biri çizim rengi olmak üzere iki renk kullanılabilir...

Yazmışım, yanlış bir bilgi olmuş. 320x200 çözünürlükteki 2 grafik ekranı titretilerek çözünürlük 640x200'e çıkarılamaz. Bunun için ekranın yarım piksel titretilmesi gereklidir, bu da durağan grafiklerde mümkün değildir. Hires-Interlace, sadece üst üste gelen renk karışımları ile renk sayısının arttırılmasını sağlayabilir. (kayan grafiklerde yarım piksel shiftingi Krestage 1 ve Krestage 2'deki 640x200 ve 640x400 scrollarda görebilirsiniz ancak bu kafanızı karıştırmasın)

Maya'da Kertenkele Yapımı

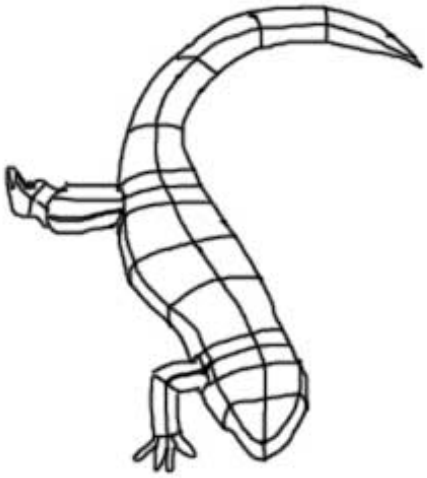
Özgür 'Lord Henry Wotton II' Yıldırım

3d Programlarında en çok hoşuma giden şey şu, istediğiniz anda istediğiniz herşeyi yapabiliyorsunuz, ama herşeyi... Bazen bir vidanjör, bazense bir şezlong. Sadece hayal etmek yeterli... Bu size de ilginç geliyorsa, birlikte kertenkele yapmaya hazırız demektir.

Şekil 1.

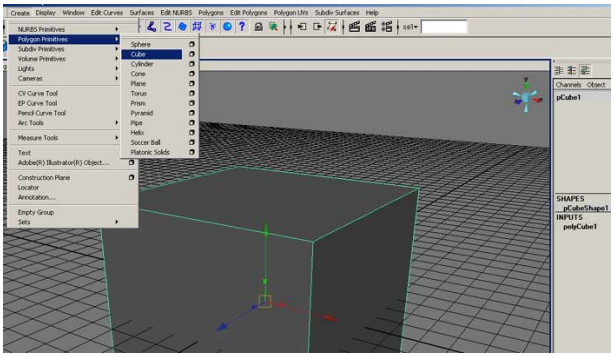
Bildiğimiz kertenkeleler. O sinsi ve içten pazarlıklı görünümlerinin aksine aslında kelebek gibi bir kalpleri vardır.

Modelleyeceğimiz cismi önce analiz etmemiz gerekiyor tabii ki. Şekli nasıl? Üç boyutta nasıl ifade edilebilir, şekli hangi temel basit şekillere benziyor, vesaire, vesaire. Bu ön hazırlık ne kadar detaylı ve programlı olursa 3d programı karşısındaki işimiz o kadar kısa ve kesin olur elbette. Dolayısıyla önce fotograftan cismin şekline basitçe bakalım.



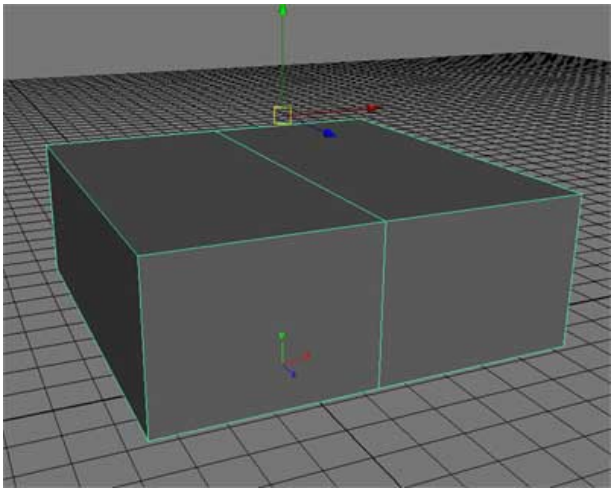
Şekil 2.

Görüldüğü gibi Kertenkelemiz aslında çok basit bir şekilde ortaya çıkarılabilecek bir geometriye sahip. En azından kabaca bunu söylemek mümkün. Gelin yavaş yavaş bu geometriyi oluşturmaya çalışalım.



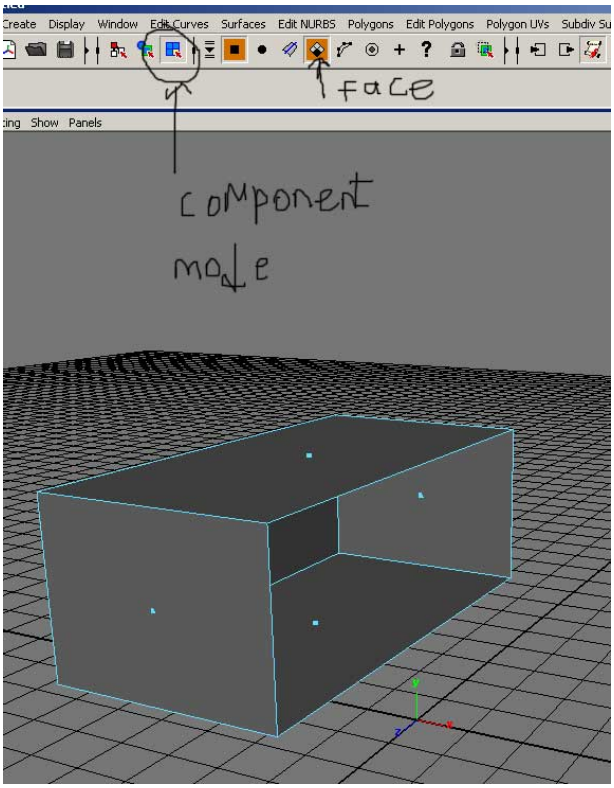
Şekil 3.

3D programlarıyla objeleri farklı metodlarda modelleyebiliriz. Bu metodlardan biri olarak ben "primitive" bir objeyi temel alarak onu tıpkı bir heykeltraş gibi işleyerek modelimi oluşturmaya çalışıcam. Yukarıdaki şekilde maya'da çok kısa bir zaman içinde kertenkeleye dönüşecek olan standart bir küp görülüyor.



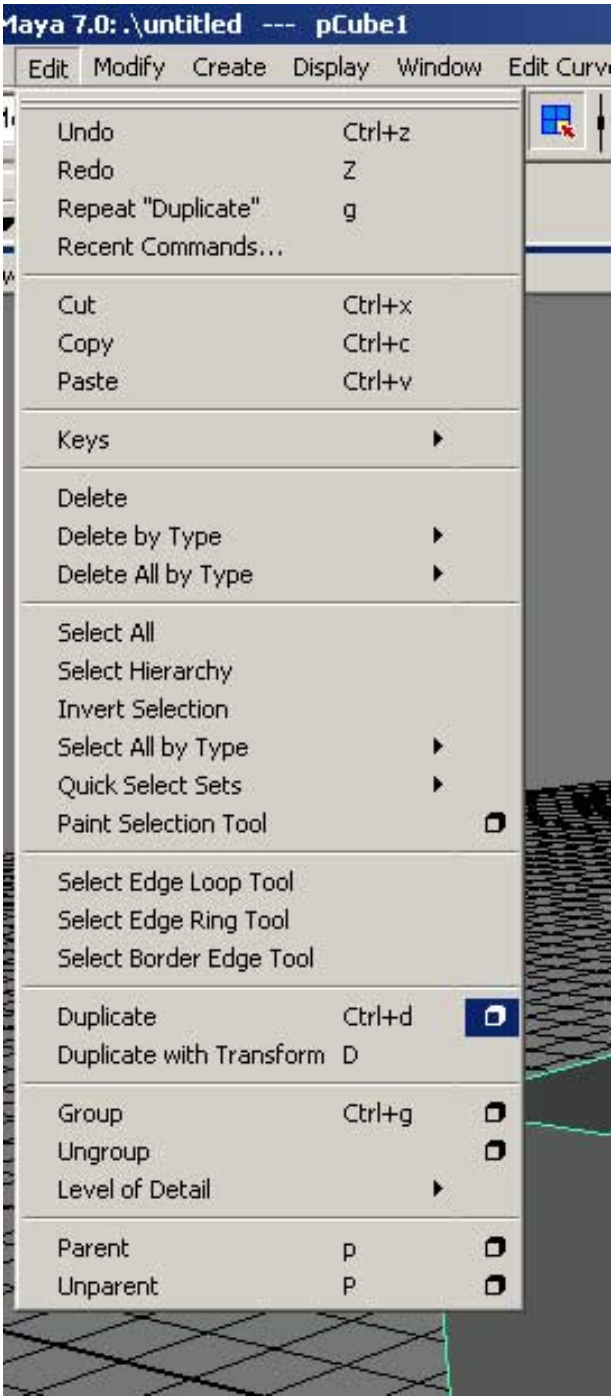
Şekil 4.

Daha sonra mayanın arayüzünde sağ tarafta kalan "input" değerlerinden subdivisions width değerini 2 olarak belirleyip kübe biraz daha "geometri" kazandırdım, ortadan bölünmüş oldu.



Şekil 5.

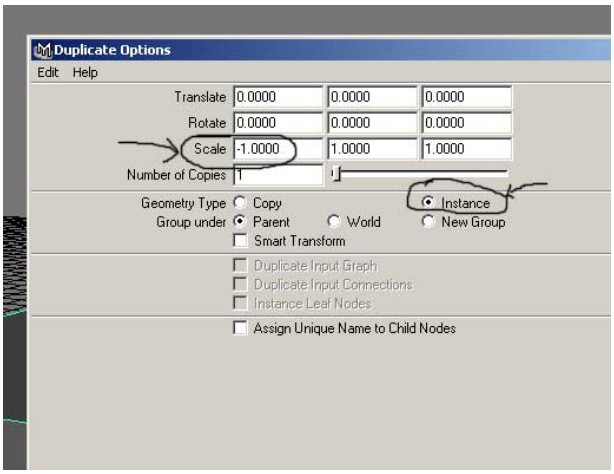
Bu arada hatırlatmak istediğim ufak bir husus var. Maya'da objeler genel olarak 2 mod' da bulunur. Obje modu ve komponent modu. Obje modunda objeyi bir bütün olarak seçip işlem yaparız, komponent yani bileşen modunda objenin geometrisini değiştirmekte kullanacağımız bileşenleri seçeriz. Vertexler, face' ler komponent mod' da seçilebilir. Yukarıdaki şekilde komponent moduna geçip kübün sağ tarafında kalan face' leri sildim, enine scale ettim ve içi boş yarım bir dikdörtgenler prizmasını elde ettim. Bunu yapmamdaki amaç kertenkeleme başlarken en iyi başlangıç objemi oluşturabilmek. Objenin yarısını sildim çünkü onu komut yardımıyla oluşturup bir tarafta yaptığım değişiklerin diğer tarafa da uygulanmasını sağlayabileceğim, mirroring de denir buna.



Şekil 6.

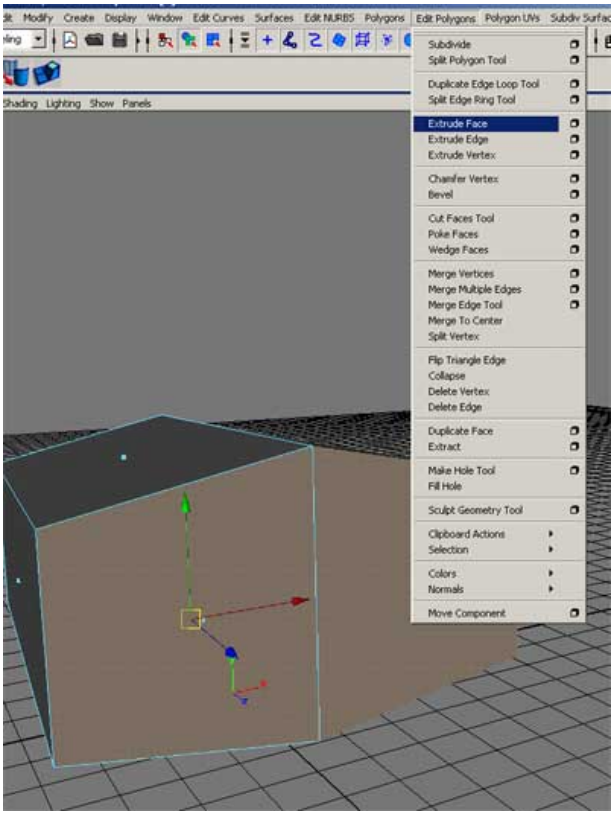
Mirroring için bir çok yöntem mevcut. Bazı yöntemler direkt "mirroring" yapıyor, hiç bizi uraştırmıyor, fakat ben bu örnekte "hardcore" gideceğim, işin zor ve uzun yolu.

Yukarıdaki foto dan gördüğümüz gibi edit menüsünden duplicate komutunun sağındaki ufak kutucuğa yani options' ların olduğu alana geliyoruz.



Şekil 7.

Şimdi özellikleri ayarlayalım. Scale x değeri -1. Çünkü sildiğimiz face' leri oluşturacağız. Geometry type instance. Çünkü orjinal geometride yaptığım değişikliklerin kopyalanmış geometride de uygulanmasını yani "instance" olmasını istiyorum.

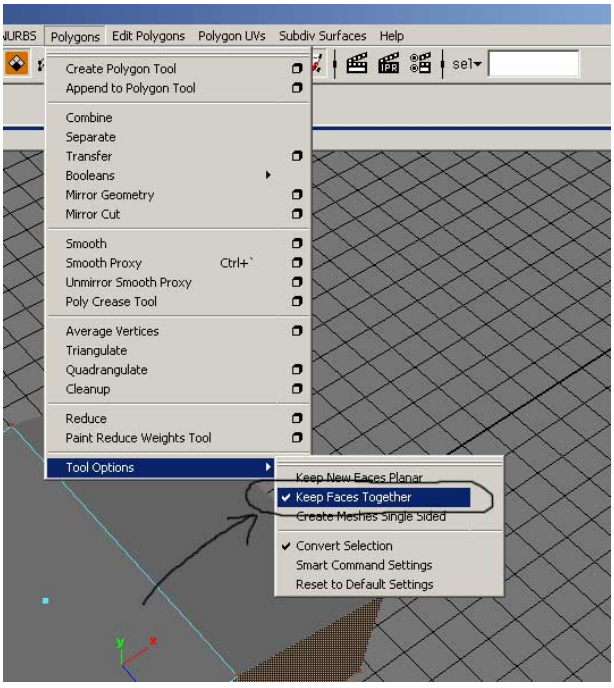


Şekil 8.

Evet şimdi objemizin eksik kısmını tamamladıktan sonra en çok kullanacağımız komuta geldi sıra. "Extruding face". Yani varolan bir face' den yeni face' ler oluşturma. Böylece kertenkelemizin eksik kısımlarını yeni geometriler oluşturarak tamamlayacağız. Şekilde kübün vertexlerinden tutarak başlangıç objeme biraz daha şekil verdim. Kertenkelemin ana gövdesinin başlangıcı olacağı için böyle bir şekille başlamak uygun. Daha sonra şekilde gördüğünüz face' i seçip onu extrude ederek ana gövdeyi oluşturmaya başlayacağız.

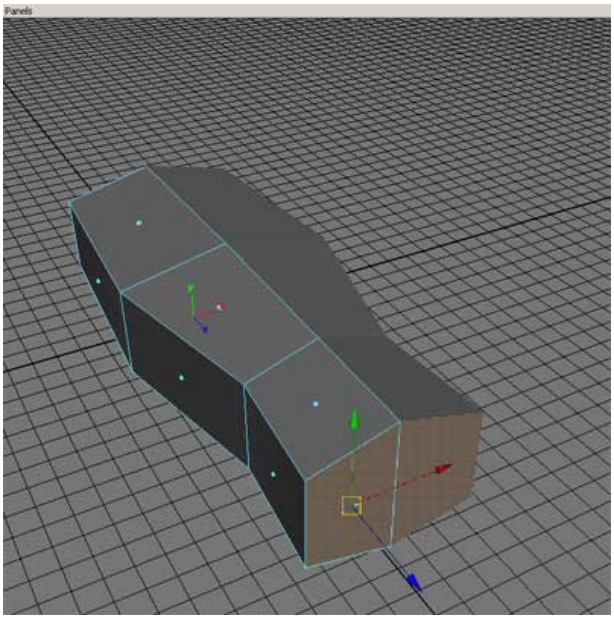
Şekil 9.

Önceki araştırmalarımızda görmüştük ki, iki ayak arasını oluşturmak için başlangıç objemizi iki kere extrude etmemiz gerekir. Dolayısıyla hemen extrude işlemlerimize başlayalım.



Şekil 10.

Unutmadan!!! Keep faces together özelliğini şekilde olduğu gibi aktif hale getirelim ki çoklu face extrude' lerinde tüm face' ler birlikte "davransın." Diğer türlü hepsi ayrı ayrı extrude edilmiş olur, bu bizim istediğimiz şey değil!

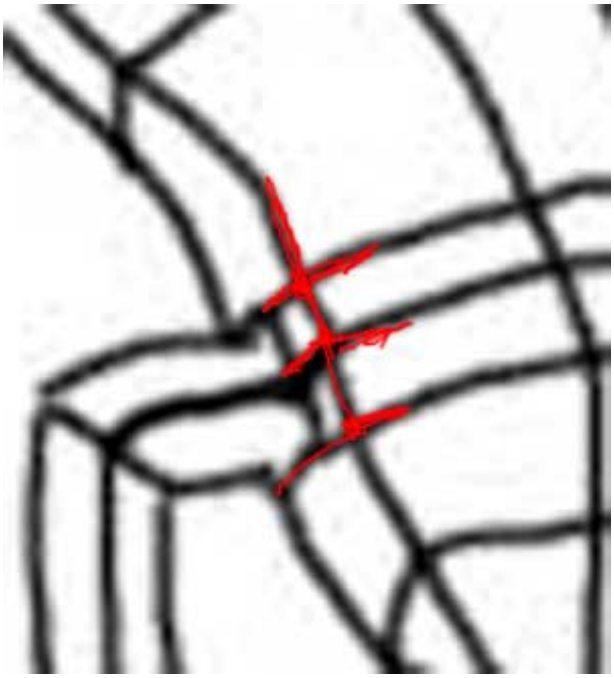


Şekil 11.

Evet iki kere extrude edip vertexleri çekiştirerek daha önce çizimini yaptığımız objede belirttiğimiz gibi geometriyi şekillendirdim. Bu metod, yani "face' i extrude et ve daha sonra vertexleri çekiştirerek istenen şekli oluştur" metodu bu kertenkeleyi oluşturmada ANA metod olacak, hem çok basit, ama aynı ölçüde işimizi görecek bir metod. Profesyonel "modeller"' lar tabii ki bu işi çok "kapsamlı" bir şekilde yaparlar, gelişmiş tool' lar kullanırlar, farklı durumlarda detaylı komutları ve araçları vardır, biz daha yumurtada civciviz. O yüzden basit ama "durumu kurtarıcı" şeyler yapmak durumundayız. Zamanla işi öğrene öğrene bizde uzman abilerimiz gibi olabiliriz. Bu tutorial' ın bir amacı da bu, sabretmeyi ve azmetmeyi bilmek.

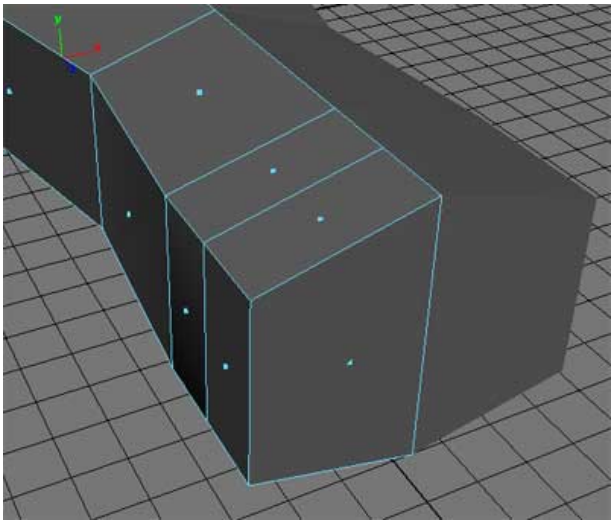
Geometri çizimimizde bacaklar ve kollar arasında üç "blok" geometri vardı ve bunları oluşturduk, şimdi kolların oluşacağı gövde geometrisine geldi sıra.

Hemen inceleyelim:



Şekil 12.

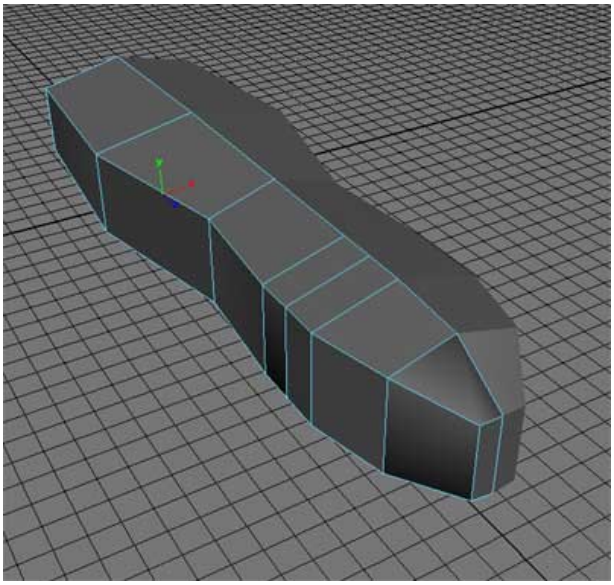
Gördüğünüz gibi içinden kolu çıkaracağımız geometriyi oluşturabilmek için 2 kere daha extrude etmek gerekecek.



Şekil 13.

Evet iki kere extrude ettim ama vertexlere dokunmadım geometriyi değiştirmedim. O yüzden geometri aynı kalacak şekilde uzadı. Şimdi sıra geldi boynu ve kafayı oluşturacak geometrilerimize..

Çizimimizde bu öğeler için 2 blok geometri gerekiyor. yani iki kere daha extrude edip vertex çekiştireceğiz.

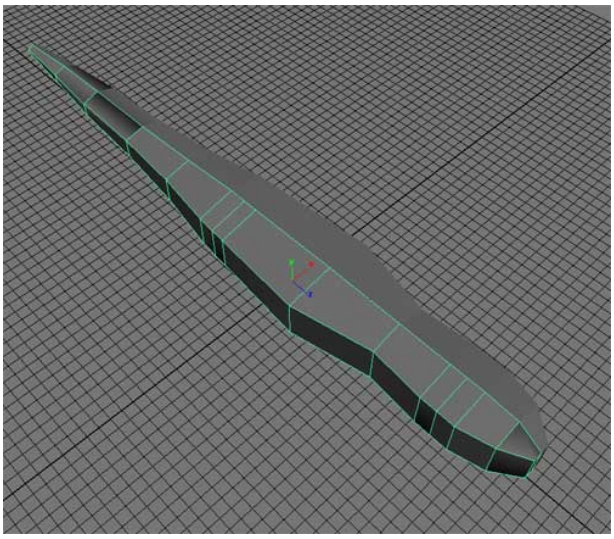


Şekil 14.

Evet 2 kere extrude ettikten sonra şekildeki gibi vertexleri yeniden konumlandırarak istediğim geometriyi elde ettim.

Arka bacak içinde 2 extrude uyguladıktan sonra, kuyruk geometrisi için çizimimde de "kabaca" belirttiğim gibi 5 extrude' e ihtiyacım var.

Hemen extude ede ede oluşturalım.

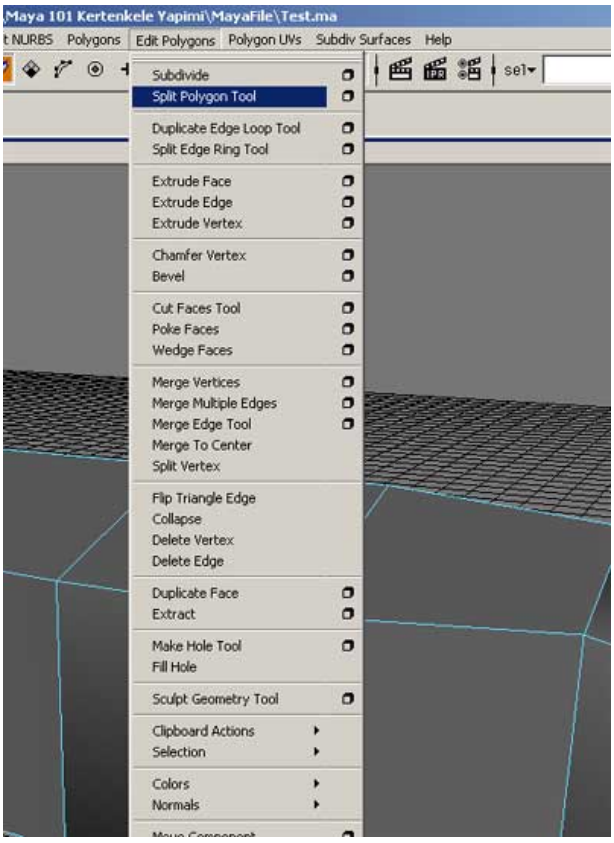


Şekil 15.

Bu arada extrude tool' u nasıl kullandığım belki hiç bilmeyen arkadaşlar tarafından merak edilebilir, tıpkı diğer programlarda olduğu gibi extrude komutunu uyguladıktan sonra W tuşuna basıp default olarak global moddaki move tool "tutamac" ından yani

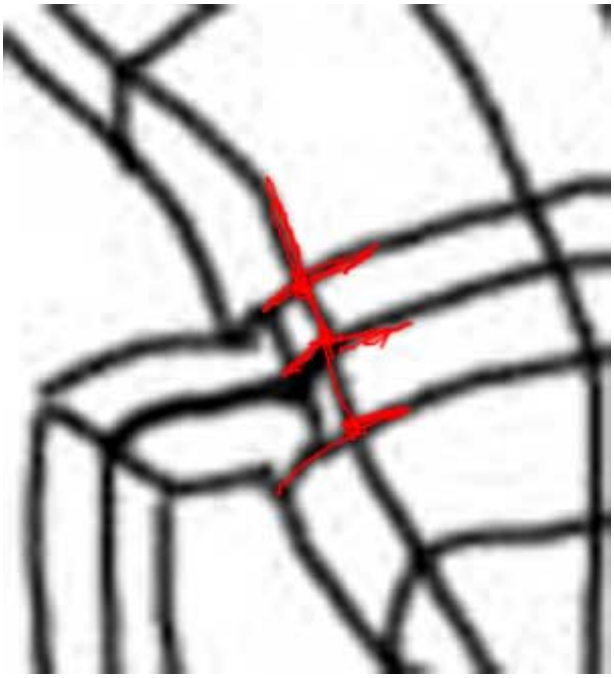
"gizmo"sundan tutarak önce belirli bir bölgeye kadar extrude edilmiş face' i taşıyorum, burada dikkat edeceğiniz nokta extrude edilmiş geometri mirroring de referans alınan çizgiyi yani orijin çizgisini geçmemeli, ki karşı tarafa doğru şekilde yansısın. Ondan sonra komponent moddan vertex moduna geçiyorum. Sonra vertexlerle oynayarak objeyi yeniden şekillendiriyorum. Vertex tweaking. Modelling is just vertex tweaking, that's all. Demiş üstad. O jilet gibi modelleri böyle mi yaparlar bilinmez (Bilmek için google' a modelling tutorial yazın yahu! :)) ama bizim yaptığımız şimdilik sadece bu. Biraz 3d programlarıyla haşır neşir olmuş kişiler hemen bu extrude işlemlerini yapar. Kesinlike çok basit işlemler bunlar. "Halkın anlayacağı" türden.

Çok mu basit geldi, o zaman bir tool daha kullanalım, split poly tool.



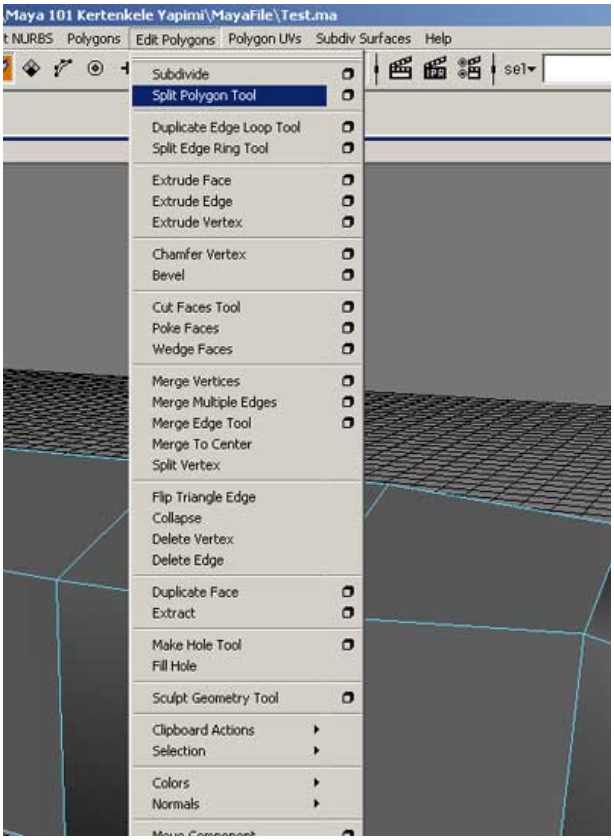
Şekil 16.

Bu tool ile ne mi yapacağız, cizimimize yakından bir daha bakalım.



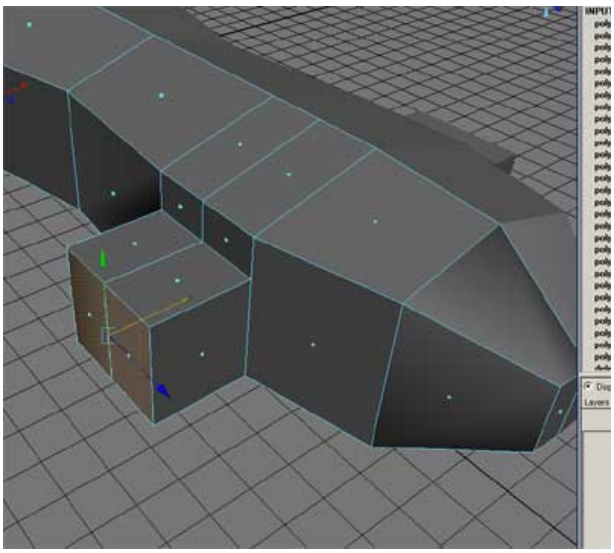
Şekil 17.

Gördüğümüz gibi kolların üstünde geometri oluşturmamız lazımmış. Diyebilirsiniz ki illa cizimimize sağdık mı kalmamız lazım, istediğinizi yapmakta serbestsiniz, ben fotosopta kertenkele fotosunun üzerinden geçerek bu çizimi oluştururken 3d geometrinin nasıl olabileceği konusunda az çok fikir sahibi oldum, yani bu kertenkele 3d' de neyi ifade eder, edge' ler nasıl olmalı, (Gevur buna edge flow der) gibi. Dolayısıyla bu çizimime sağdık kalarak kertenkelemi oluşturuyorum, bu benim işimi 3d programının karşısında çalışırken çok rahatlatıyor, ne yapmam gerektiği belli. Ne yapacağım? Tabii ki bu geometriyi oluşturmak için ilgili face' i "split poly tool" ile keseceğim:



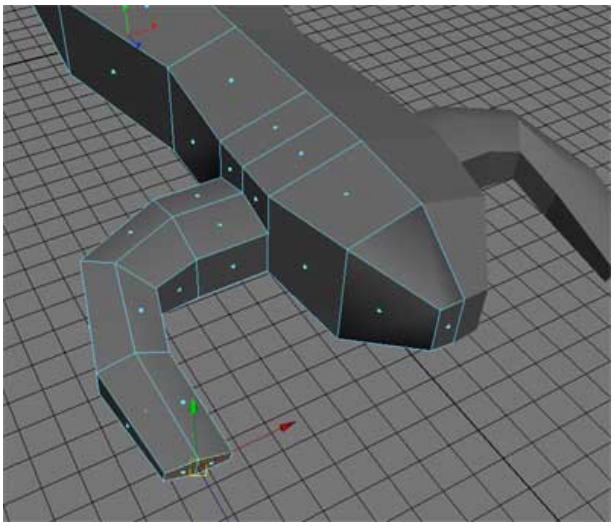
Şekil 18.

Evet. Kestim, <enter> a basarak tool' u sonlandırdım. Split tool ile işim bitti. Sıra geldi bu yeni oluşan geometriden kolları oluşturmaya, bunun için emektar tool' um extrude face' i elime alıyorum:



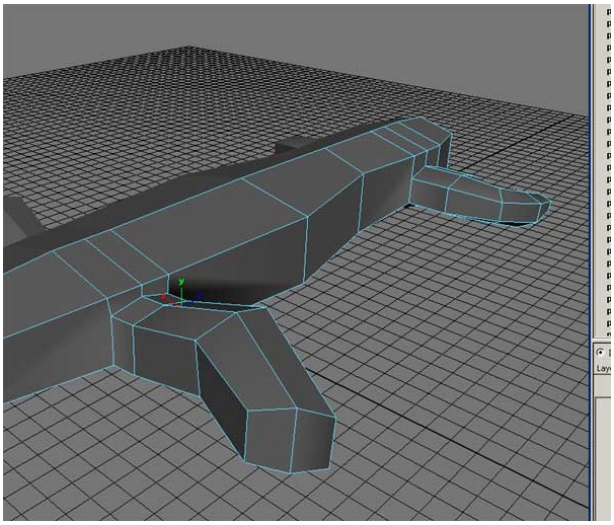
Şekil 19.

Ilk hamleyi yaptım. Çizimime göre 2 extrude' de kolu tamamlamam lazım ama 2 extrude yetmeyecek gibi. Daha fazla extrude uygulamam lazım ki smooth ve detaylı bir geometri oluşsun.



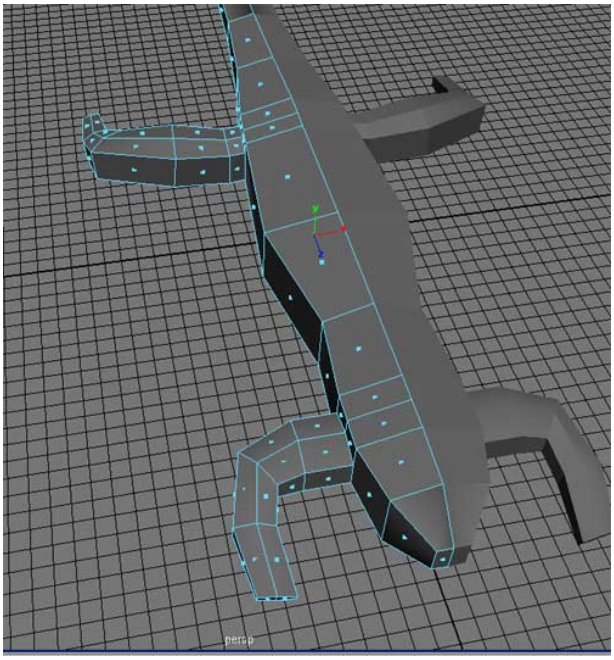
Şekil 20.

Evet. 4 kere extrude ederek vertexleri çekiştirdim ve kertenkele koluna benzeyecek şekle getirdim. Sıra geldi bacaklarıda aynı bunun gibi oluşturmaya:



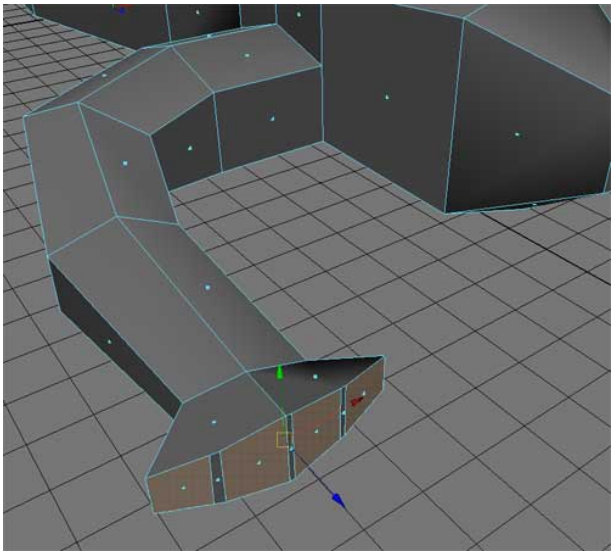
Şekil 21.

Önce bacağın çıkacağı face' i kestim, sonra extrude' ler uyguladım ve vertexleri çekiştirdim. Şöyle bir şey çıktı ortaya:



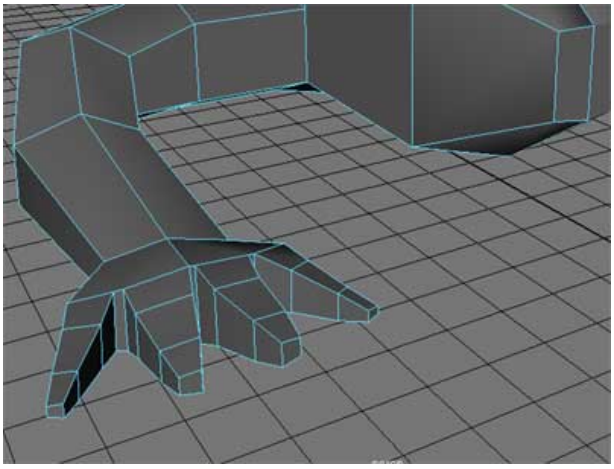
Şekil 22.

Sıra geldi, elleri ve ayakları oluşturmaya. Aynı metod, Face' i split edip gerekli geometriyi oluştur, extrude et, vertexleri düzelt, extrude et, vertexleri düzelt.



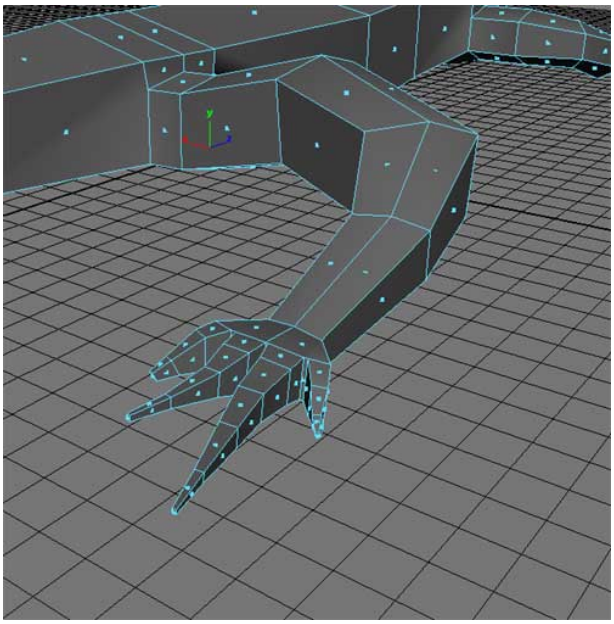
Şekil 23.

4 üncü extrude' dan sonra:



Şekil 24.

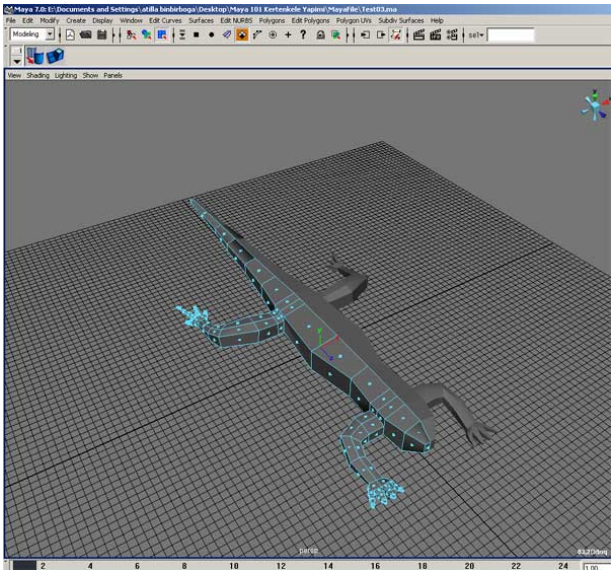
Aynı şekilde ayakları ele alalım.



Şekil 25.

Hıncal Uluç çok bilmişliğiyle "Böyle ayak mı olurmuş kardeşim!" demeyin, valla kertenkele ayağı böyle.

Sonuçta en son geldiğimiz nokta şu:

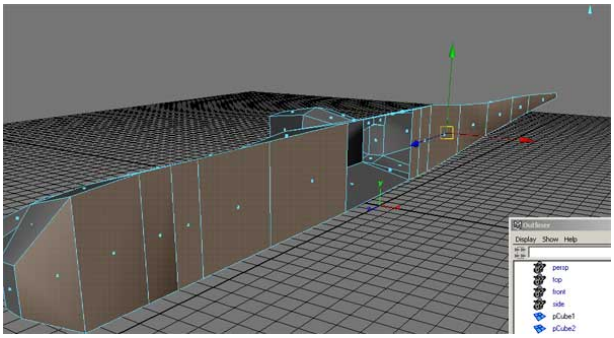


Şekil 26.

Modelleme aşamamız bitti. Geçmiş olsun. Sıra geldi kopya tarafı orjinal taraf ile birleştirmeye. Resmi olarak ikisi farklı objeler şu an için. Dolayısıyla birleştirmek lazım...

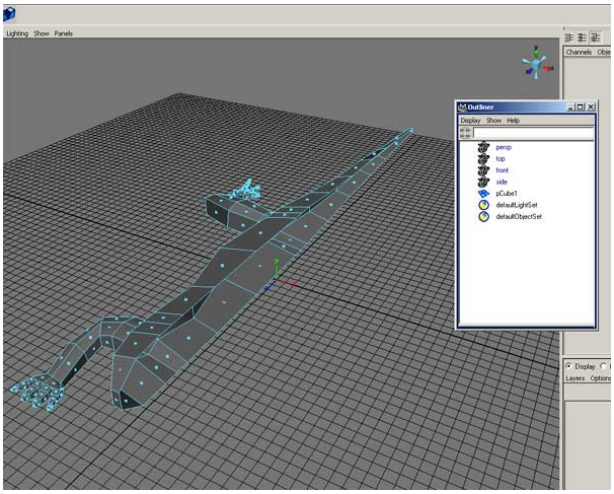
Önce kopya olan kısmı komple silelim, bunun için Windows>Outliner ' i açın ve kopya olan geometriyi seçin, default olarak pCube2 dir onun ismi. Sildikten sonra ilginç bir manzara ile karşılacaksınız, kertenkelenin içindede geometri oluşmuş. Çünkü aslında biz yarım bir geometride işlem yaptık, dolayısıyla içe doğruda geometri oluştu. Tüm bu gereksiz ve hatta zararlı faceleri seçip delete tuşuyla siliyoruz.

Seç:



Şekil 27.

Sil:



Şekil 28.

şimdi okumaya devam et:

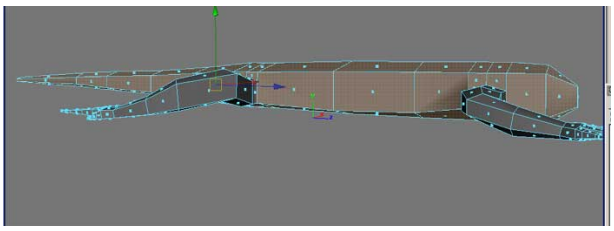
Eksik kısmı tamamlamadan önce gelin isterseniz önce texturing meselesini halledelim, buradaki avantaj şu, yarım objelerle texturing de tıpkı modellemede olduğu gibi sadece yarım kısma UV layout oluşturmak yeterli, dolayısıyla yine işin yarısından kurtulmuş olacağız. Nesneyi UV layout' uyla mirrorladığımızda UV layout' da mirror edilecek.

UV layout hazırlarken, bazısı bunu software' lerin "auto projection"' larına bırakıyor, otomatik olarak çıkan UV set' e ufak müdahaleler ile sonlandırıyorlar. Ben bu işi manuel olarak yapacağım. Genel olarak objenin basit bir geometrisi yoksa otomatik projeksiyonlamayı tercih etmem. Maneul olarak hazırlamak size çok daha fazla imkan verir. Ve çoğu zaman aslında işleri daha da kolaylaştırır. Şunu unutmayın ki UV mapping başlı başına bir uzmanlık alanı ve o tertemiz UV layoutları profesyoneller çok detaylı projeksiyonlamalarla ve "UV editing"' le çıkartıyorlar.

Projeksiyonlama için, gövdeyi ayrı, kolu ayrı ve bacağı ayrı ayrı projeksiyonlayacağım, böylece 3d bir nesnenin tüm yüzeyleri 2d bir ortama (UV) en iyi şekilde nasıl aktarılır buna uraş vermeye çalışacağız.

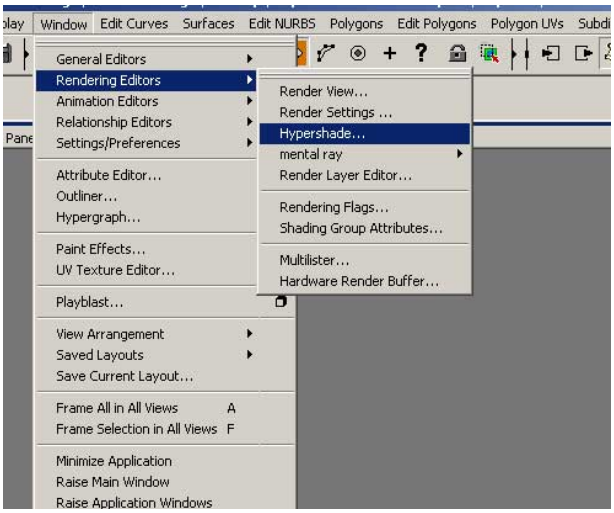
Önce sadece yarım gövdeyi seçin, kolun ve bacağın herhangi bir komponentinin "select" durumda olmadığına emin olun.

Seçilmiş gövde:



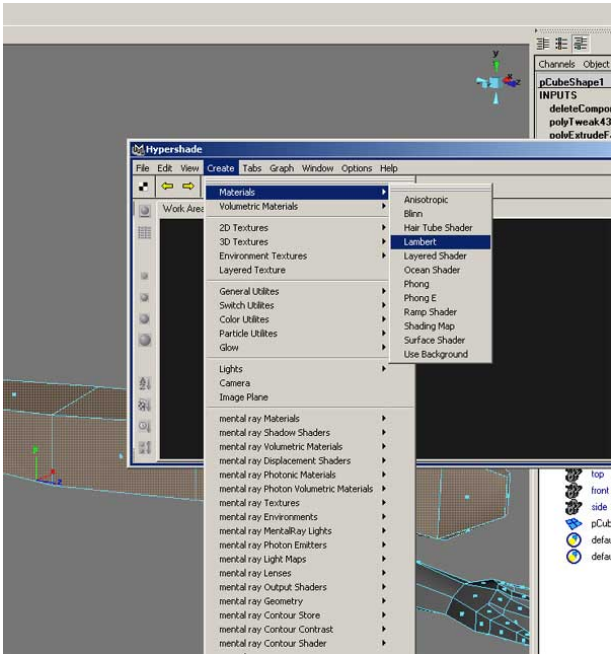
Şekil 29.

Önce "Hypershade"' i yani Maya' nın "material" editorünü açıp gövdeye ayrı bir materyal atayalım:



Şekil 30.

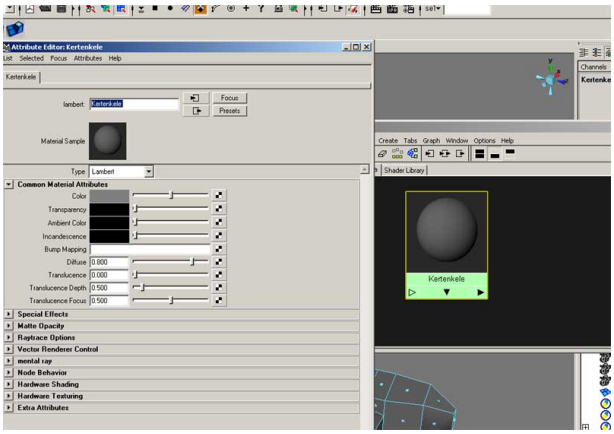
Default bir lambert shader işimizi görecektir:



Şekil 31.

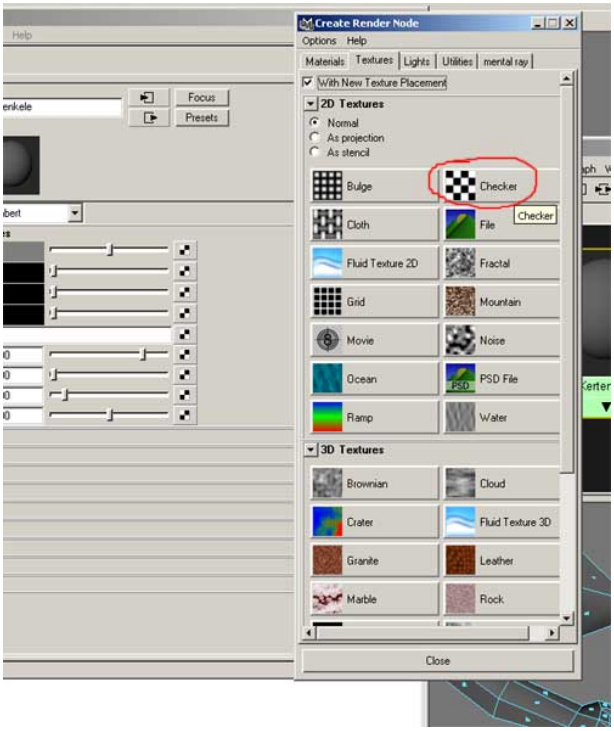
Lambert materyalimize ad verelim, ben kertenkele dedim. Ve kertenkele' ye bir texture atayalım, bundaki amaç şu, UV layout' u hazırlarken hali hazırda texture' lu nesneyi referans olarak kullanıyoruz ki, layout ne ölçüde orjinal texture' ü "distort" edecek, hangi geometriye ne kadar texture alanı düşecek önceden görüp ona göre UV' imizi editliyoruz.

Bunun için Kertenkele material' imizin Attribute editor' ünü çift tıklayarak açıyoruz,(Nesnelerin özelliklerini gördüğümüz ctrl+a kısayollu genel bir editör) Color kanalına gelip oradaki siyah beyaz kareli tuşa basıyoruz.



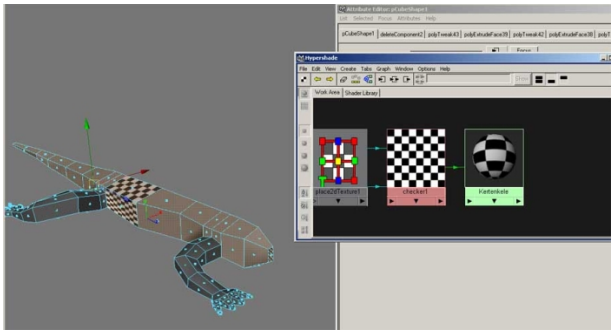
Şekil 32.

Create Render Node penceresi açılması lazım, oradan checker' ı seçiyoruz. Checker Texture artistleri için klasiktir, UV mapping' de distort olacak alanları çok iyi göstermesi gibi nedenlerle genelde Checker' la UV "check" edilir.



Şekil 33.

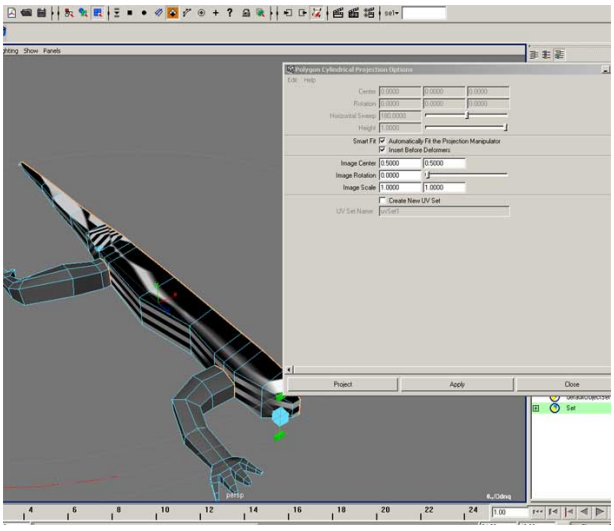
Ve Checker' in atanmış hali:



Şekil 34.

Şekilde de gördüğümüz gibi Texture geometrinin bazı yerlerinde mevcutken bazı yerlerinde yok. Bunun nedeni şu, objeyi oluştururken primitive bir objeyle, yani standart küple başladık, her default primitive objenin hazır bir UV layout' u vardır. Bizim kübümüzünde bir zamanlar böyle bir özelliği vardı. Fakat biz geometriyi extrude ederek değiştirip kertenkele şekline sokarken aynı zamanda hali hazırdaki Küp UV layout' ununda ırzına geçtik. Maya' da her editlenmiş primitive' den sonra UV' leriniz uçar gider. (Çünkü artık o mayanin bildiği bir küp değildir!) UV'lerinizi Yeniden oluşturmak durumundasınız.

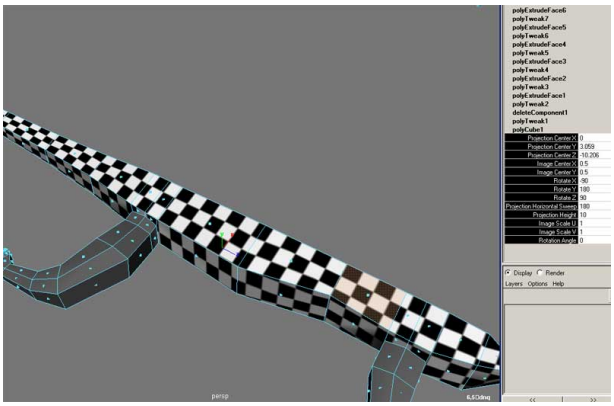
Bunun için seçili gövdemize projeksiyon uygulamamız gerekecek:



Şekil 35.

Polygon UVs menüsünden Cylindrical projection' u uyguluyoruz. Neden planar veya spherical değil? Nedeni çok basit, komplike bir geometrimiz var ancak bu geometri hangi primitif şekle daha daha çok benziyor, buna karar verip ilgili projeksiyonu uyguluyoruz, bu örnekte yarım gövde yine yarım bir silindire benziyordu.

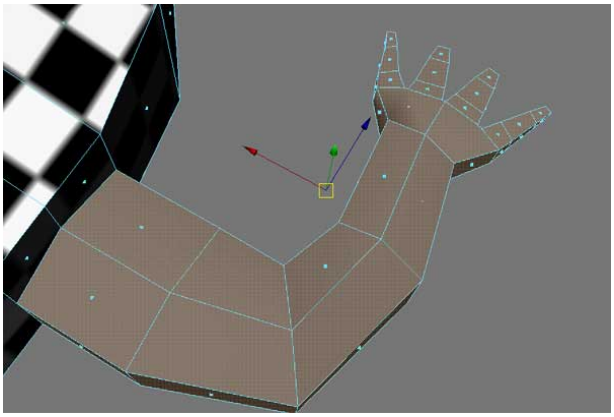
yukarıdaki resimdede gördüğümüz gibi projeksiyonladığımızda texture' ümüz yine istediğimiz gibi gözükmedi. Nedeni projeksiyonun konumu ve açısının yanlış olmasında. Projeksiyonu uyguladıktan sonra bunu değiştireceğiz. Bunun için projektörün manipulator tool' unu kullanarak (projeksiyonu uyguladığınızda default olarak seçilir, deselect ederseniz klavyenizdeki T tuşuyla manipulator tool' a geri dönün. Ya da benim gibi bu işi ekranın sağındaki alanda bulunan "Channel Box" isimli bölgede ilgili özellikleri sayıyla girerek en uygun açıyı ve konumu bulun.



Şekil 36.

Yukaridaki resimde sağda seçili olan değerlere dikkat! büyük ihtimal yakın değerleri bulacaksınız.

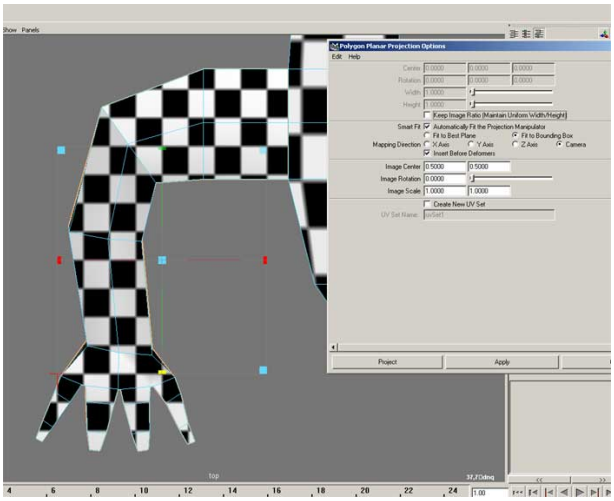
Sıra geldi kollara. Kolları klasik olarak üst kısım ve alt kısım olarak iki ayrı projektorle projeksiyonluyorum. Üst kısmı yanlar dahil seçiyorum:



Şekil 37.

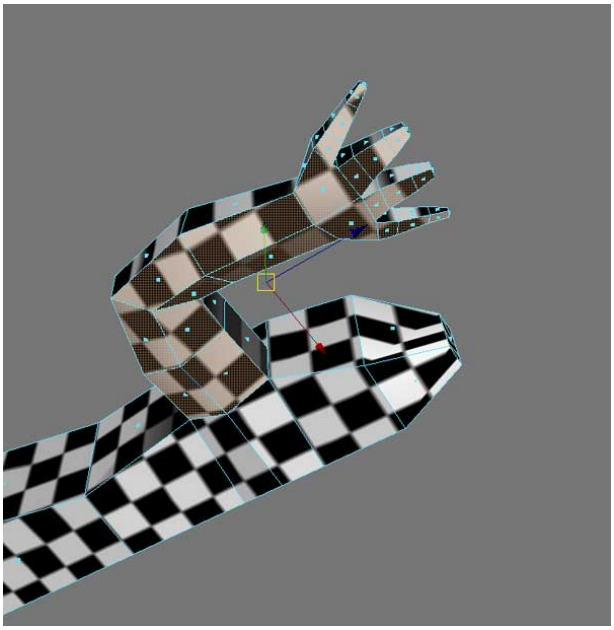
Ve cylindrical projection uyguluyorum. Silindirik çünkü yanları da aldığım için bunu tercih ettim.

Alt kısmı seçiyorum ve bu kez Planar! projeksiyon uyguluyorum, çünkü geometri oldukça flat.



Şekil 38.

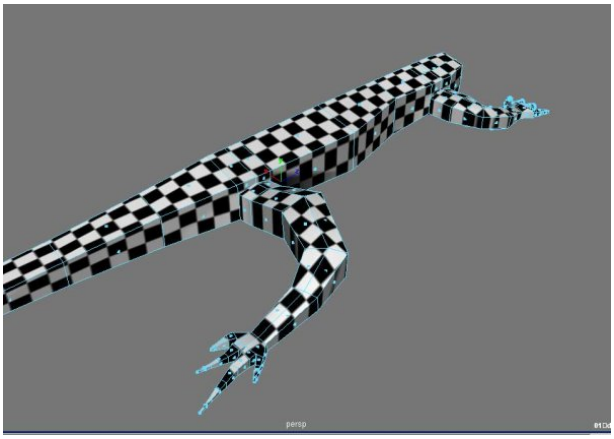
Şekilde de görüldüğü gibi kollardaki kareler vucuda göre daha küçük, bunun anlamı şu. Kollarda vucuda göre daha fazla texture alanı içerecek, yani çözünürlükten nasibini en fazla bu alan alacak. bunu istiyor muyum, şu an hayır, kare boyutunu vucuttakilerle aynı hale getirmek için projeksiyondaki "projection height and width" değerleriyle oynuyorum.



Şekil 39.

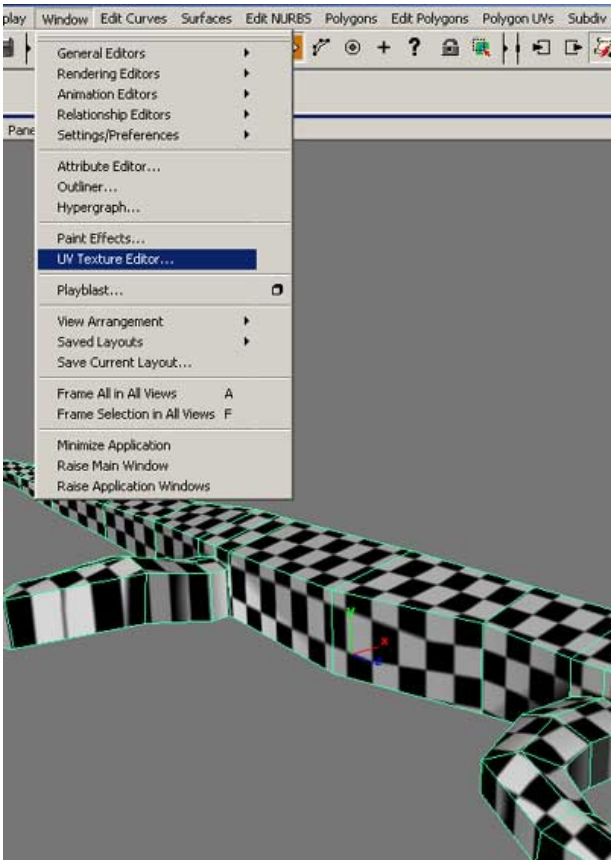
Kare büyüklükleri birbirine benzemeye başladı. Bu iyi.

Aynı şekilde bacağıda üst ve alt olmak üzere projeksiyonluyorum:



Şekil 40.

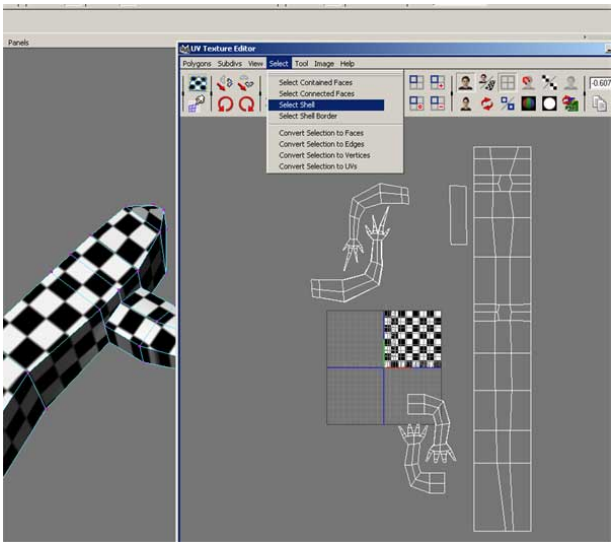
Ve projeksiyonlama ile işim bitiyor. Şimdi Windows> texture editor' e gelip biraz UV editing yapma zamanı geldi!



Şekil 41.

Şimdi texture editörü açtığınızda karmakarışık bir sahneyle karşılaşabilirsiniz, UV' ler birbirine geçmiş olabilir. Bunun nedeni, her projeksiyon uygulandığında, projection edilen geometrinin

UV'si UV layout' un merkezinde oluştulur. Dolayısıyla tüm UV' lerimiz merkezde şu an. Bunları kolayca birbirinden ayıracağız şimdi. Texture editördeki Select menüsünden "Select Shell" komutunu aklınızda tutun. Ve editörün üzerinde sağ mouse tuşuna basılı tutarak çıkacak olan "Marking menu" den UV' yi seçin, daha sonra editörün üzerindeki herhangi bir UV' yi seçin, sadece bir tanesini, sonra select' ten select shell komutunu uygulayın, ta ta ta tamm! Sadece bir tane UV seçerek projeksiyon edilmiş tüm ilgili UV' leri seçmiş olduk, böyleyece tüm UV setleri birbinden ayırın, alttaki şekildeki gibi:

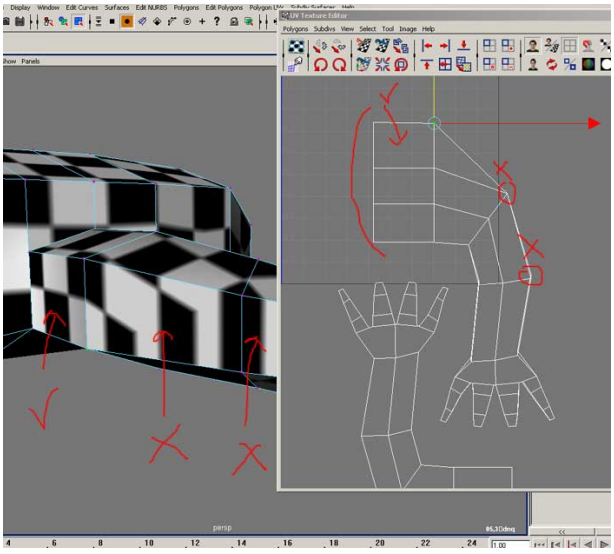


Şekil 42.

Sıra geldi distort olan yerleri fix' lemeye. Bunun için ilk önce "overlapping UV"' leri buluyoruz. Evrensel bir kural olarak aklınızda bulunsun, UV'ler hiç bir zaman üst üste gelmemeli! Amacın doğasına aykırı zaten, geometriyi 2 boyutlu bir zemine açmak, dolayısıyla tüm UV noktalarını texture editörde rahatlıkla görüp anlayabilmeliyiz.

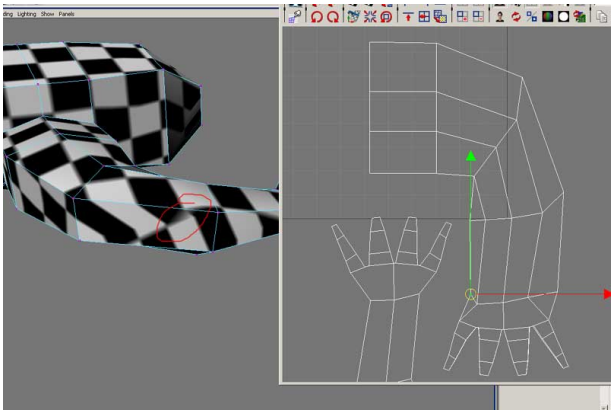
Overlapping UV' leri hemen daha projeksiyon uygularken kollarda oluşacağını tahmin etmiştim, koldaki checker' da da zira aynı belirtiler mevcut idi, çünkü oradaki checker fazlasıyla strech olmuştu.

UV' leri fixlemeye başladım:



Şekil 43.

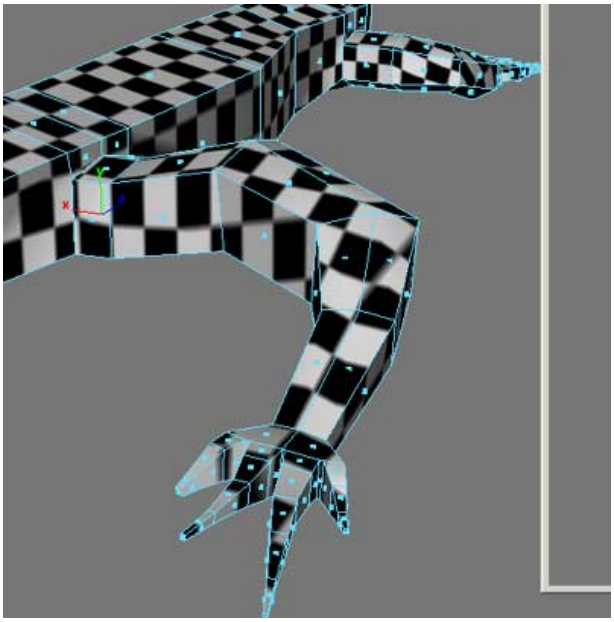
Yukarıdaki şekilde de gördüğünüz gibi ilk UV' imi editleredit-lemez checker' da dramatik bir değişiklik oldu, kareler düzeldi, texture rahatladı. Diğer editlenmemiş UV' lerdeki checker tam anlamıyla sıçıyor. Zaten bu noktaları texture editörde goremiyoruz çünkü başka UV' lerle üstüste durumda. Bunun nedeni şu, bu kola aslında iki projeksiyon az, daha detaya inmek istiyor. Ama biz hemencecik projeksiyon işini halletmek istediğimizden bu duruma katlanmak zorundayız, yani texture editörden UV' leri çekiştireceğiz.



Şekil 44.

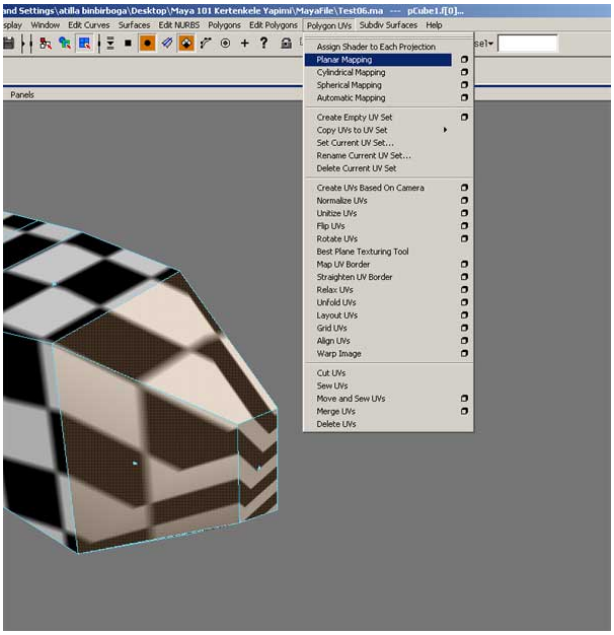
Yukarıdaki şekilde gördüğünüz gibi UV' leri tek tek editleyerek açıyorum, Şekilde gösterdiğim bölgedeki gibi blurry alanlara dikkat. UV' leri çekiştirerek checker karelerini netleştirmeye çalışıyoruz.

Bacaklara geçiyoruz:



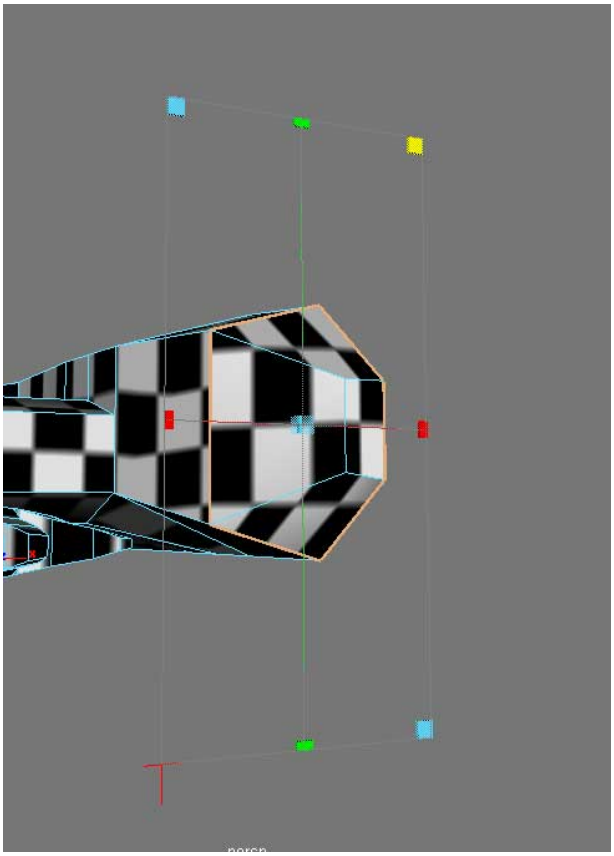
Şekil 45.

Govdeye kafayla beraber projeksiyon uygulamıştık ama ben kafayı gövdenin UV set' inden extract etmek istiyorum, basitçe kafa alanını seçip yeni bir Planar projeksiyon uyguluyorum:



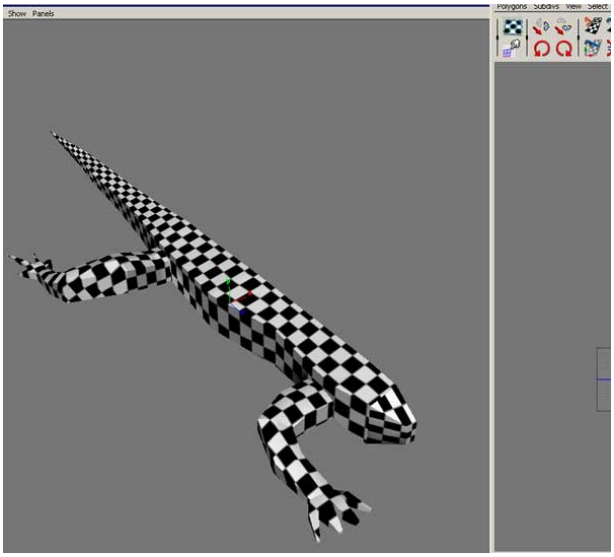
Şekil 46.

Checker karelerinin büyüklüğünü ayarlıyorum:



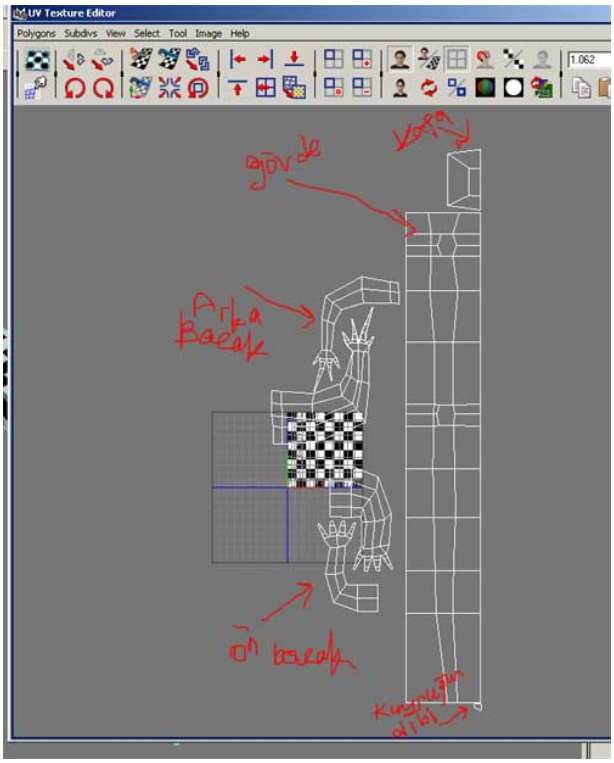
Şekil 47.

Checker' imizin genel görüntüsü:



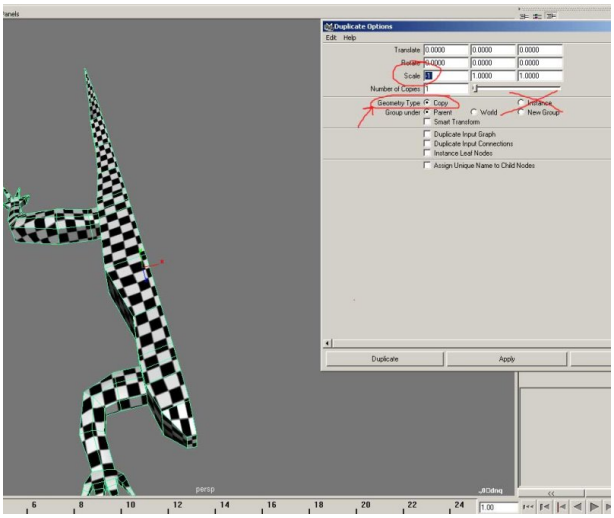
Şekil 48.

Layout' umuzun genel görüntüsü:



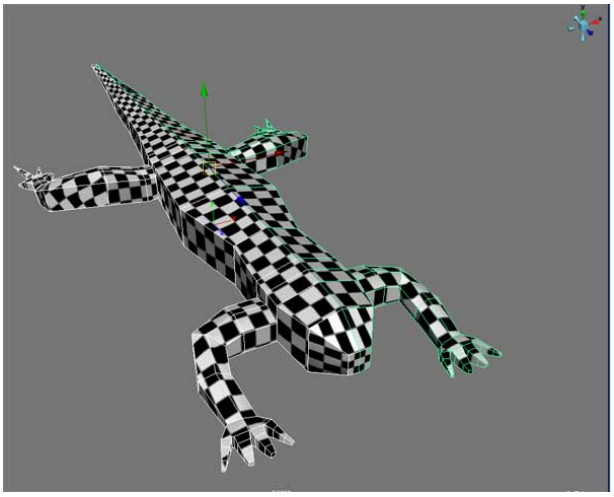
Şekil 49.

Şimdi kertenkelemizin diğer yarısını tamamlayabiliriz. Bunun için yine objeyi duplicate ediyoruz ancak! farklı parametreler ile:



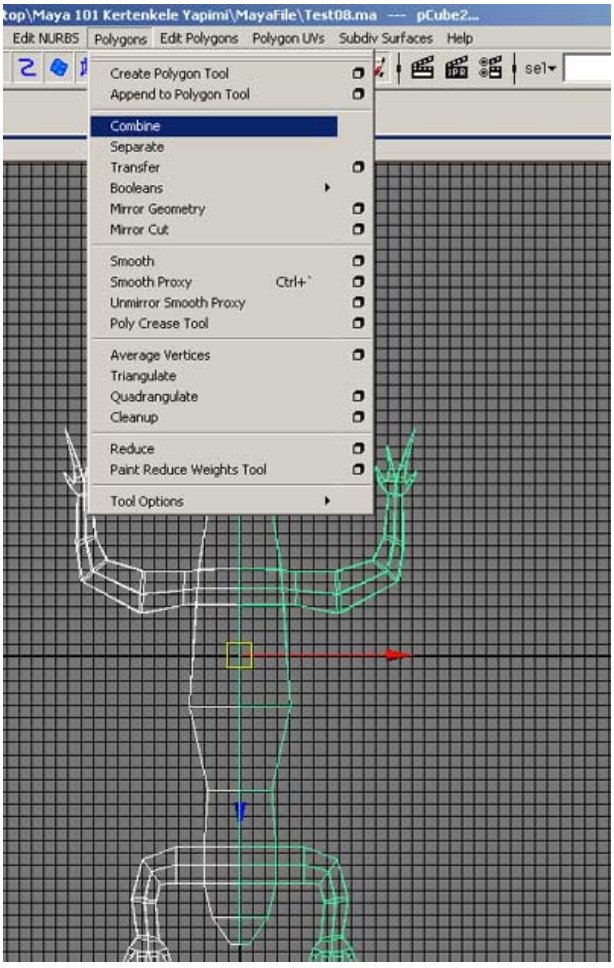
Şekil 50.

Bu kez instance değil sadece copy dedim. Birleşmiş hali şöyle:



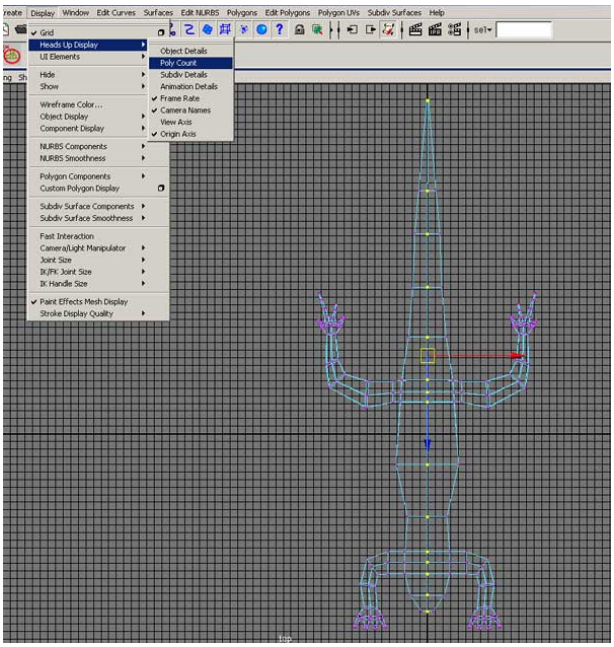
Şekil 51.

Fakat bu iki taraf hala ayrı iki obje. Combine etmemiz lazım ki tek bir geometriye dönüşsünler:



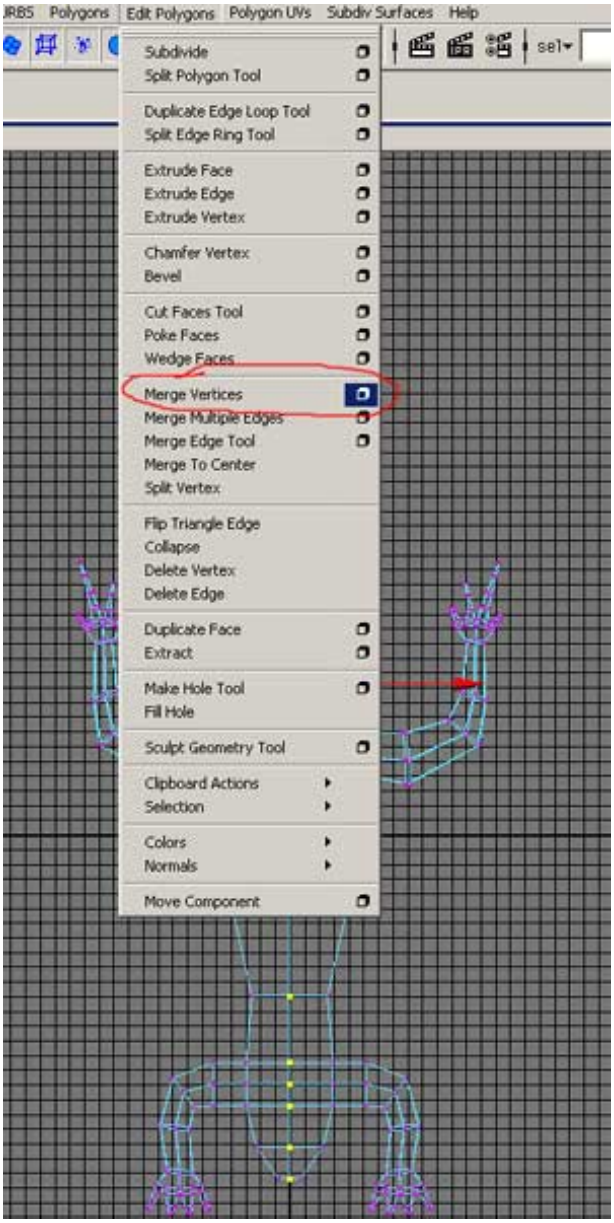
Şekil 52.

Fakat bir problemimiz daha var, bunlar tek bir obje oldu ancak "component" seviyesinde değil. Objeyinin ortasında kalan o kesişen vertexleri seçiyoruz, kontrol etmek içinde display> heads up display> poly count' u aktif hale getiriyoruz:



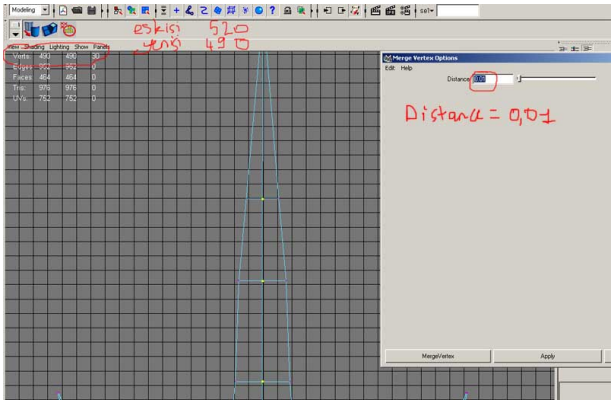
Şekil 53.

Ve seçtiğimiz vertexleri merge etmek için merge vertices komutunun option bolumune gidiyoruz:



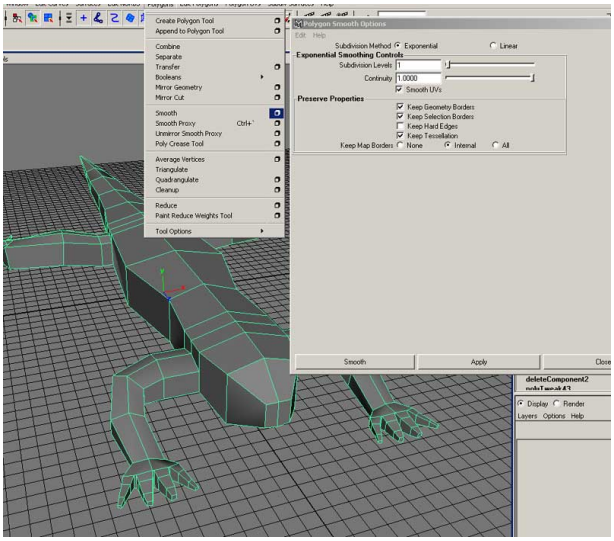
Şekil 54.

Opsiyonlarda distance değerini 0.01 yapiyoruz. Anlamı şu, uzaklığı 0.01 ve altı olan vertex' leri merge et, diğerlerini bırak. Bizim çakışan vertexlerin uzaklıklari teorik olarak 0 olduğu için merge olacak:



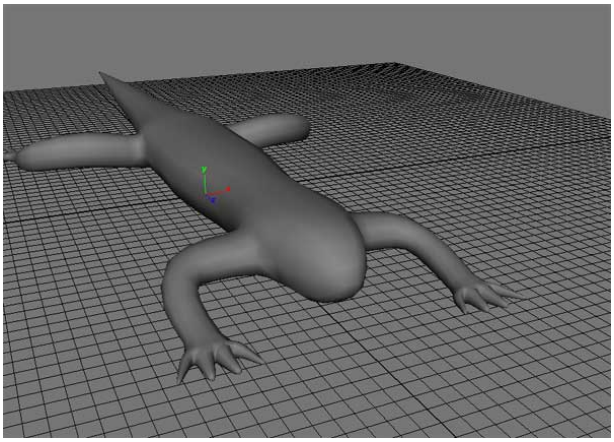
Şekil 55.

Bu arada, kertenkelemiz gayet low poly, yani hakkaten fazla LOW POLY. Çünkü çok basit bir geometri oluşturduk ve detayı fazla yok. Normalde low poly modellemede pek rastlanmasa da bu objeye bi smooth poly uygulamak istiyorum ki, alışık olduğumuz low poly nesnelere benzesin (!)



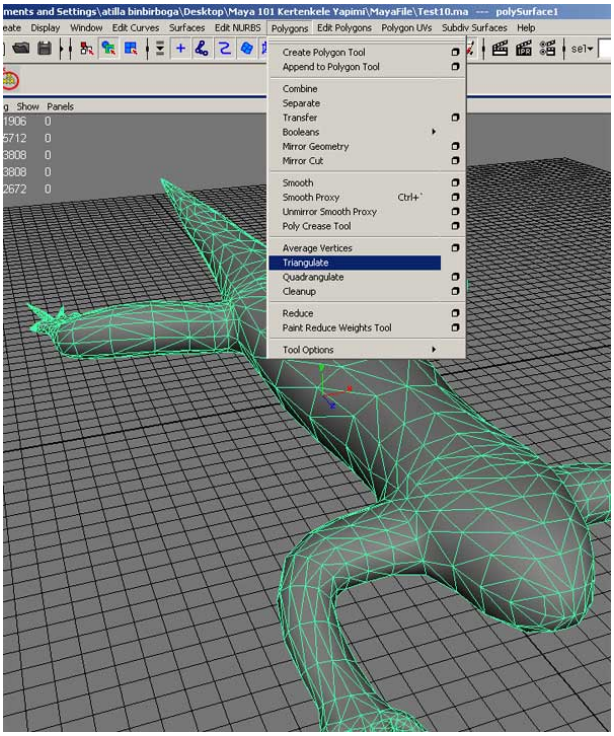
Şekil 56.

Smooth' dan sonra:



Şekil 57.

Birde polygonları triangulate ettimiydik:



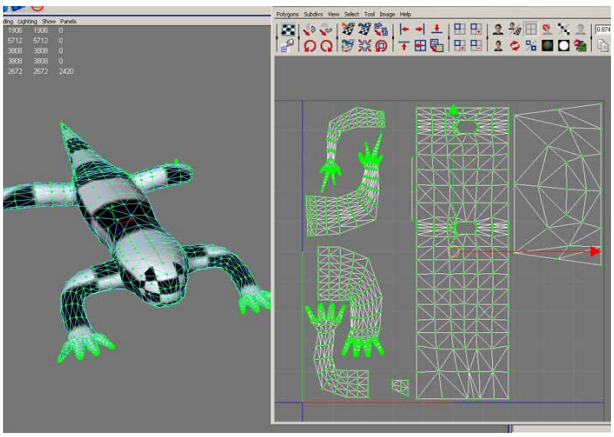
Şekil 58.

Bu arada UV' lere son kez ayarlamak istiyorum. Bir fikrim var. Kertenkelemizi gayet basit bir şekilde modellediğimizden dolayı kafa ve yüz detaylarını modellememiştik. Bunun yerine buraya bir fotoğraf koymak istiyorum. Evet! onun fotoğrafını!

Bush' un tabii ki :) Bush' tan hoşlanmama trendine uygunda olmuş olur böylece.

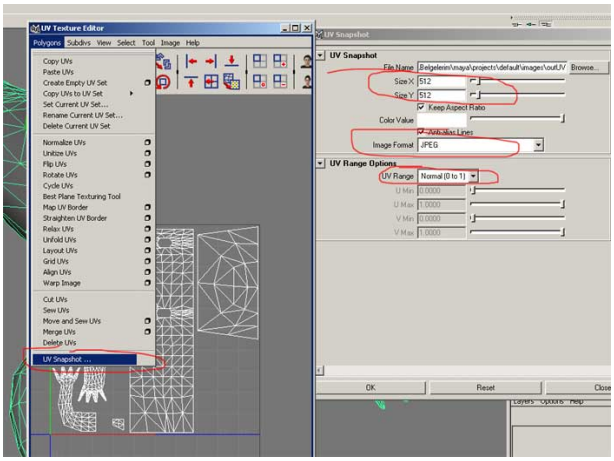
Bunun için kafa UV' lerini scale ederek texture' dan fazla yer

kapmasını sağlıyorum. Ve tüm UV layout' u texture editördeki 0-1 Aralığına yerleştiriyorum. Normalde maya ile çalışırken UV range fazla önem teşkil etmeyebilir, çünkü UV range' i istediğiniz gibi belirlersiniz, ancak başka programlara export import yaparken, UV range' i 0-1 aralığında tutmakta fayda var.



Şekil 59.

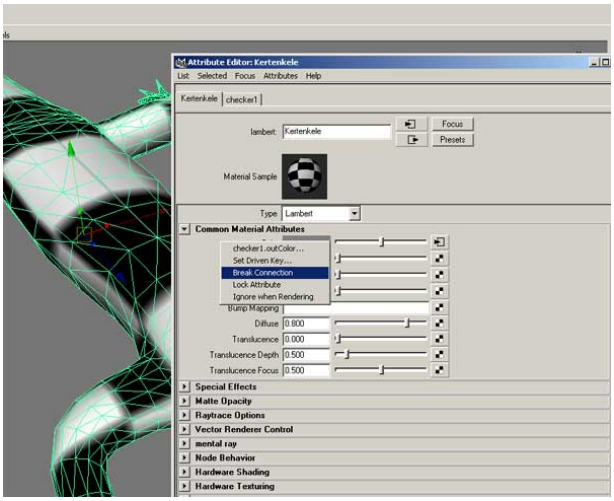
UV layout' umu texturing için almak istiyorum, dolayısıyla snapshot alıyorum:



Şekil 60.

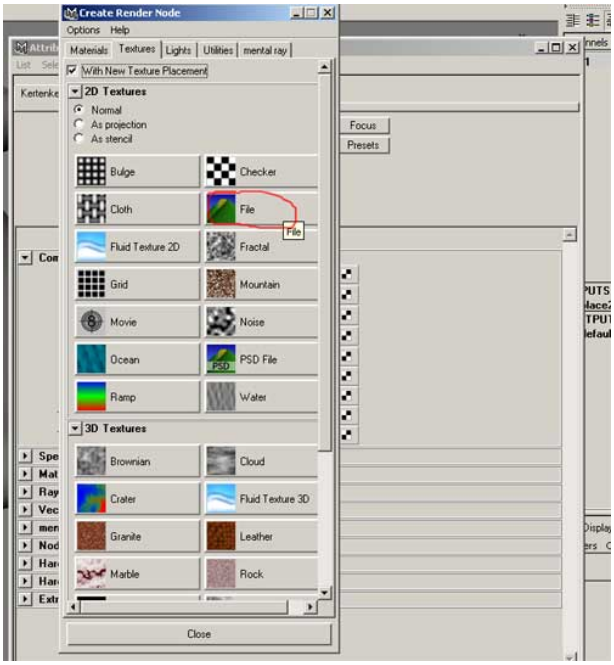
Normalde texture artistler Photoshop kullanır. Endüstri standardı. Ama ben bu işi de büyük ölçüde mayanın içinde halletmek istiyorum. Dolayısıyla Maya' nın Paint FX isimli müthiş özelliğini kullanacağım.

Önce UV layout' u texture olarak kullanacağım. Dolayısıyla aldığım snapshot' u kertenkelemin color kanalına texture olarak atayacağım, önce tabii ki Checker' dan kurtulmam lazım, color kanalında sağ tıklayıp break connection diyorum:



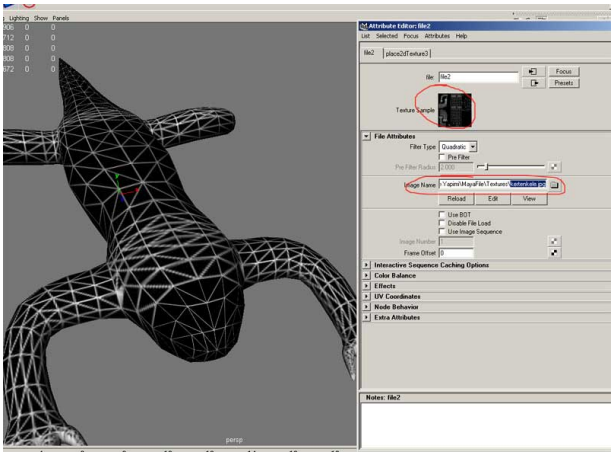
Şekil 61.

Color kanalının yanindaki siyahlı beyazlı ufak karelere tekrar basarak create render node penceresini açıp bu kez "file" diyorum :



Şekil 62.

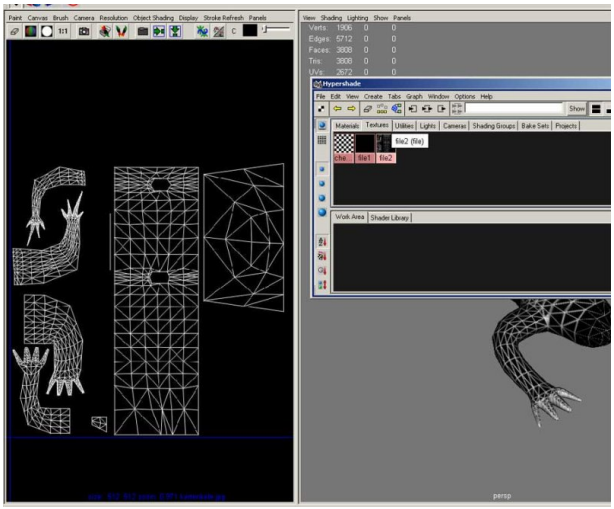
Snapshot' umun yerini gösteriyorum ve:



Şekil 63.

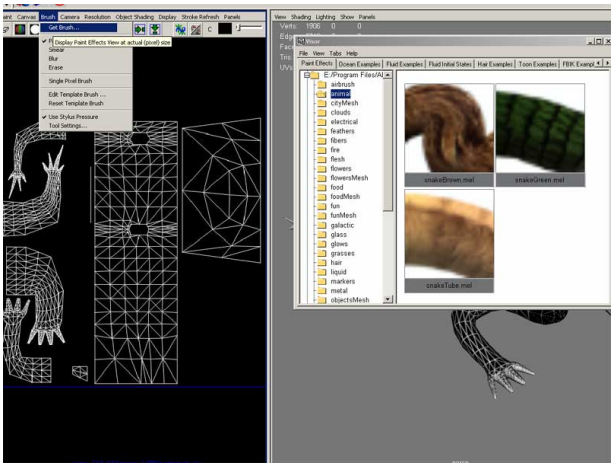
Daha sonra maya UI layout' umu Paint FX - Perspective şeklinde ayarlıyorum. Bunun sebebi, paint FX' te boyamamı yaparken perspective penceresinde direkt olarak yaptıklarımı görebileyim. Bunun için mesela windows>saved layouts' a gidin ve persp / Outliner' i seçin. Sonra outliner penceresinden Panels>Panel> Paint Effects' i seçin, default olarak Pers - Paint FX preset' inin olmaması şaşırtıcı gerçekten. Isterseniz bu UI' yi save edebilirsiniz. Ben daha önceden kaydetmiştim, ehehehe.

Daha sonra Hypershade' i açıp snapshot' umuzu PaintFX penceresine sürüklüyoruz, ki texture' müzü paint fx' te editleyebilelim. Sürükle bırak Maya'da çok onemli işlevleri olan bir husustur.



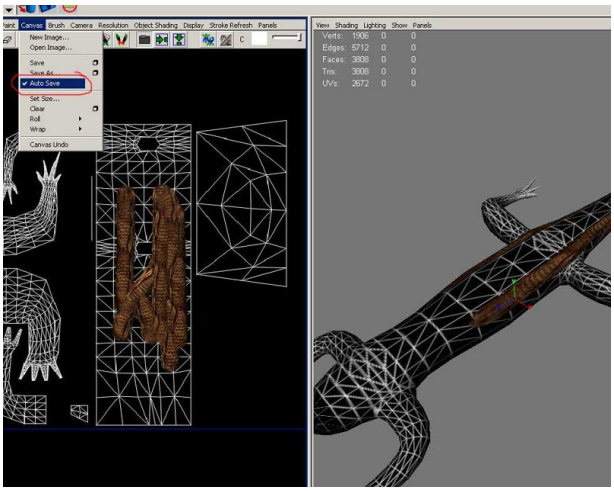
Şekil 64.

Hemen yukarıdaki Brush menüsünden get brush diyelim ve açılan visor penceresinden brush' imizi seçelim. Benim kişisel favorim snakebrown, isteyen snakegreen' i kullanabilir:



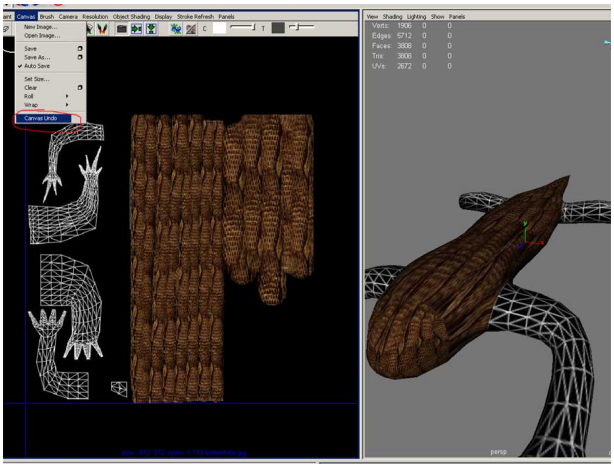
Şekil 65.

Canvas menüsünden auto save' i aktif hale getiriyoruz ki yaptıklarımız anında texture' a kaydolsun ve perspektif penceresinde texture refresh olsun:



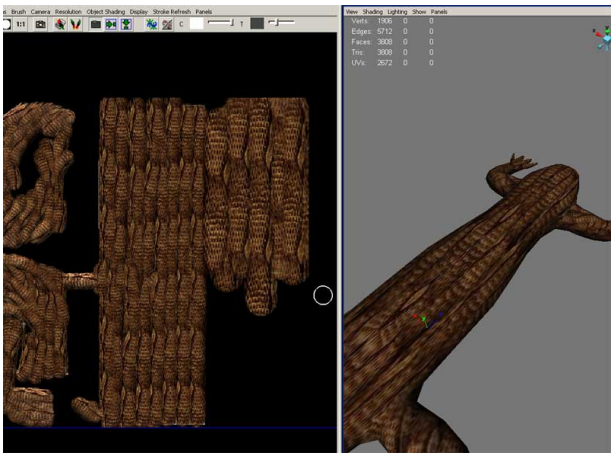
Şekil 66.

Eğer boyarken hata yaparsanız standart undo bir işe yaramaz, onun yerine canvas menusundeki canvas undo' yu kullanacaksınız, dikkat edin undo history' si sadece tek hamleyi kapsar. (Bu özelliğinden de nefret ederim)



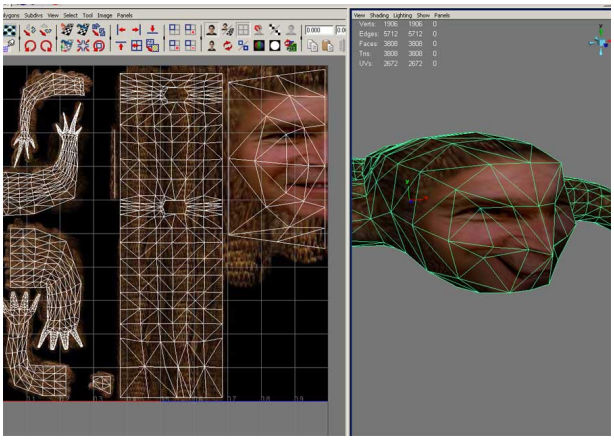
Şekil 67.

Texture' lü hali:



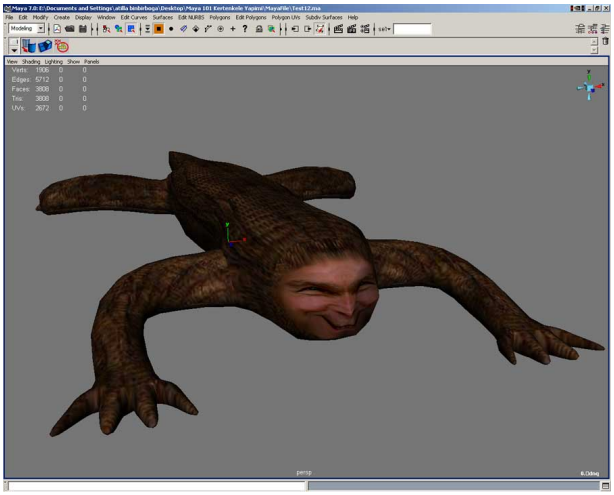
Şekil 68.

Bush' un yüz fotoğrafının texture' e uygulanması için hiç uraşmayıp direkt fotosop' u açtım ve ekledim:



Şekil 69.

Onden bi gorunus:

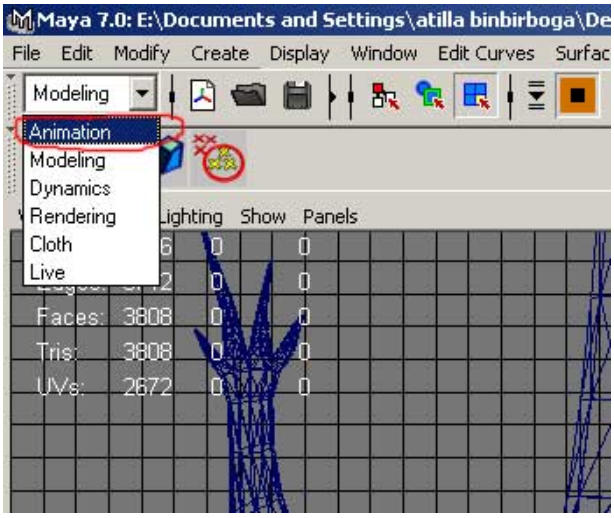


Şekil 70.

Yüz kısmında texture' u biraz daha iyi göstermek için yüzdeki vertexleri biraz çekiştirip yüze uygun bir geometri oluşturmaya çalıştım:

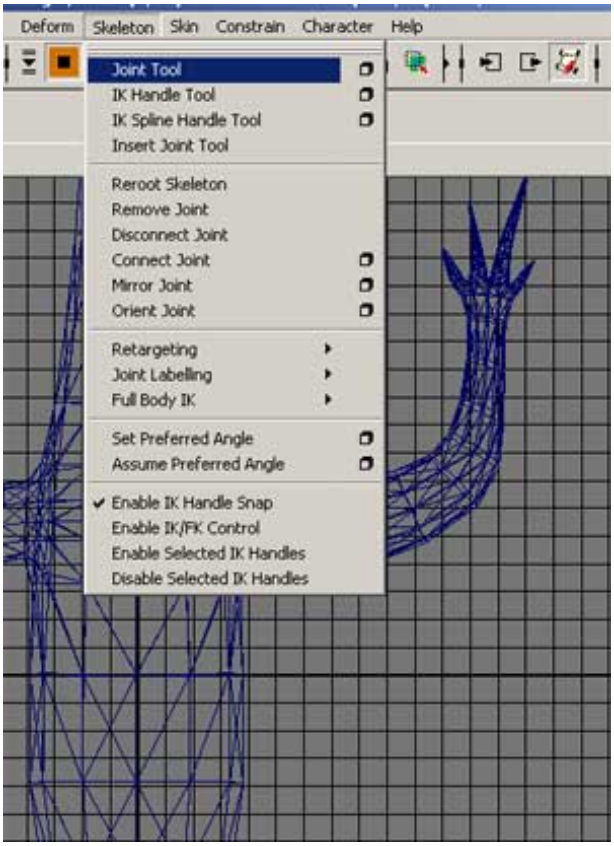
Şekil 71.

Gelelim animasyona:



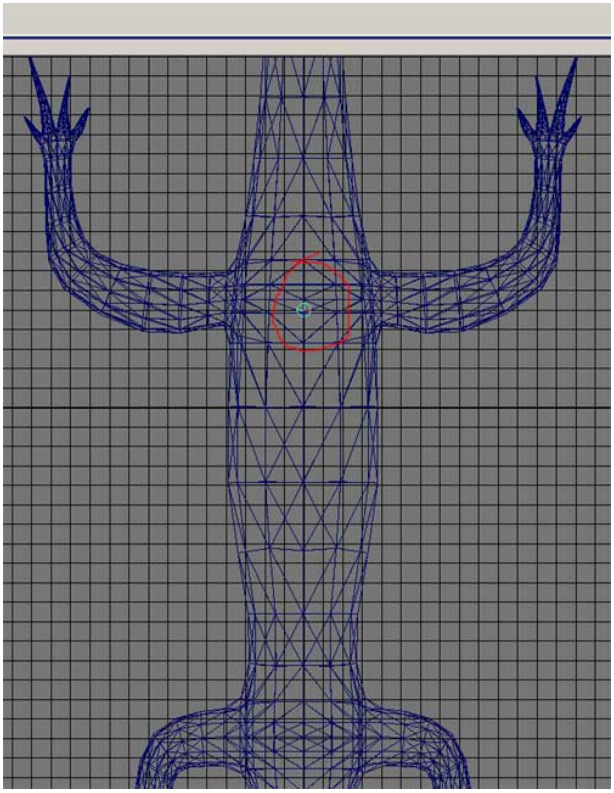
Şekil 72.

Skeleton> Joint tool' umuzu alalım elimize ve animasyon için gerekli joint (Bone) ları oluşturalım:



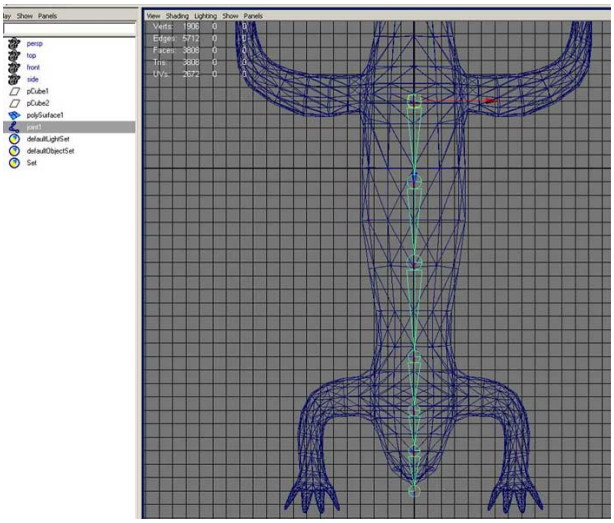
Şekil 73.

Karekterimizin iskeletini oluşturmaya "Hips" ten yani kalçadan başlıyorum. Sizde Kertenkelemizin kalçasına doğru Joint Tool' u yavaşça tıklatın.



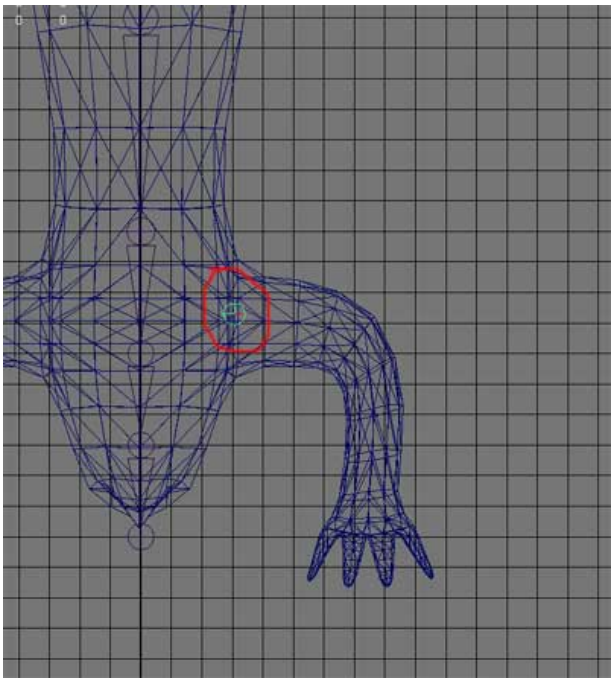
Şekil 74.

Kalçadan kafaya doğru seri olarak jointleri oluşturmaya başlıyorum. Kalça joint' i hariç altı hamlede işi sonlandırıyorum.



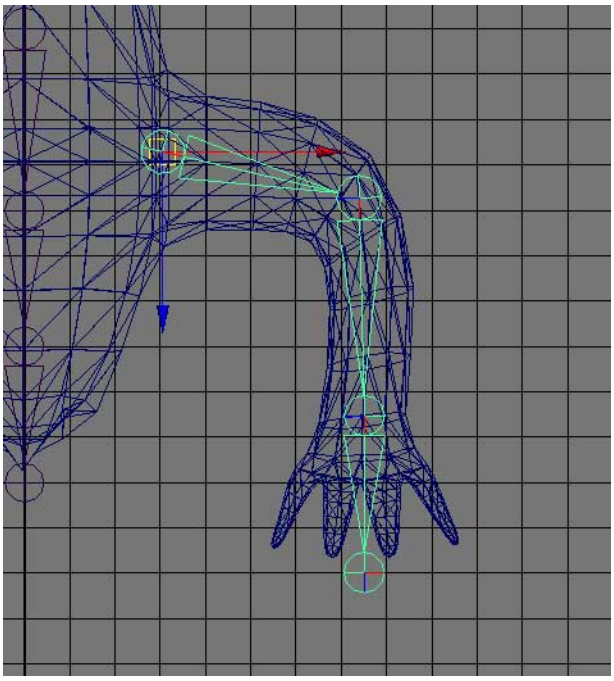
Şekil 75.

Kol joint'lerine geçiyorum:



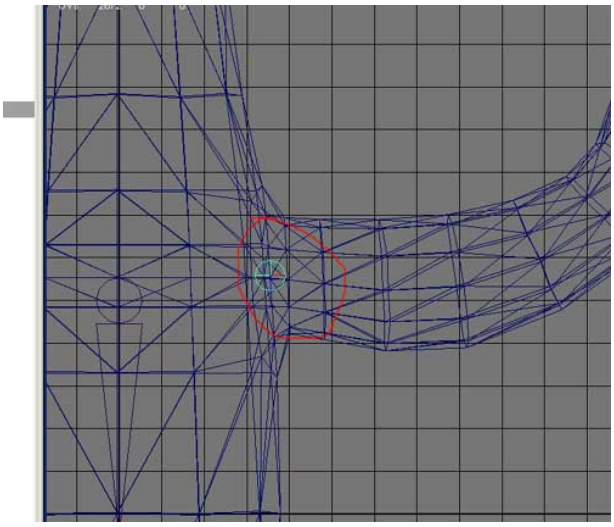
Şekil 76.

Kolu bitiriyorum:



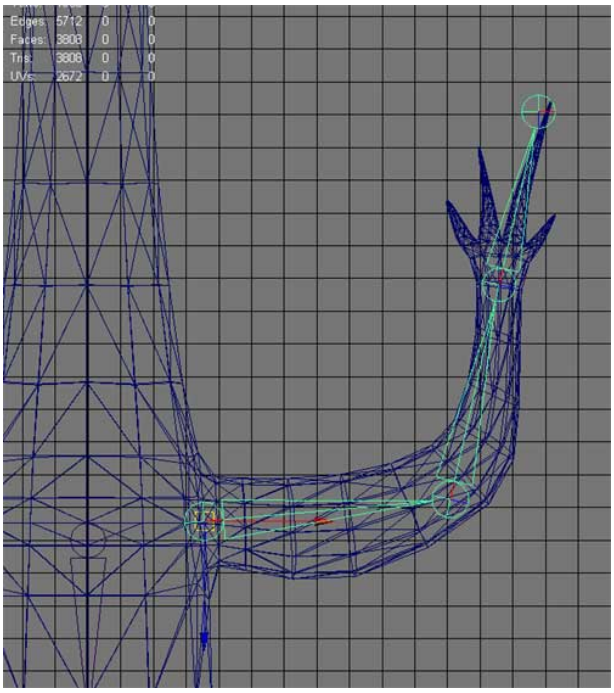
Şekil 77.

Bacaklara geçiyorum:



Şekil 78.

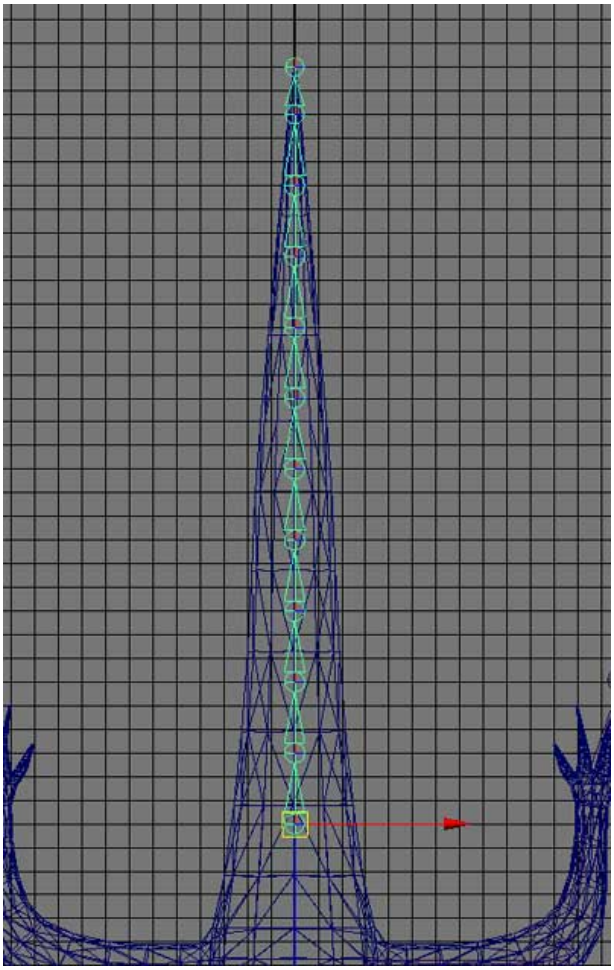
Bacağın bitmiş hali:



Şekil 79.

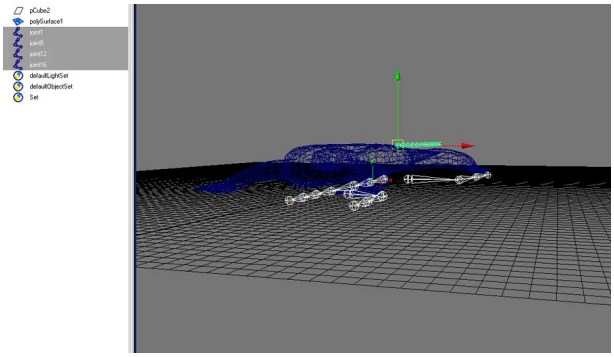
Sıra geldi kuyruğa, kuyrukta soft bir deformasyon istediğimden fazla sayıda joint kullanacağım.

Işte kuyruğumuz:



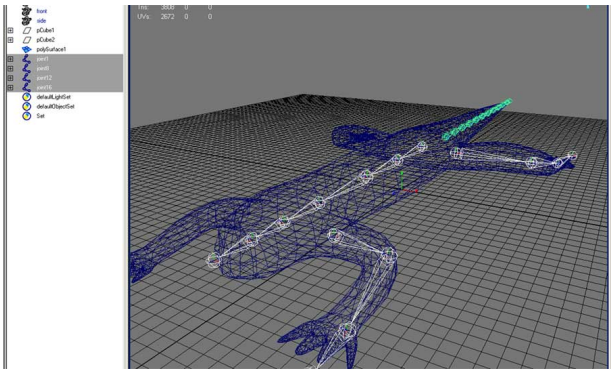
Şekil 80.

Sıra geldi joint'leri geometrimizle hizalamaya. Sadece üstten görünüşte jointlerimizi oluşturduğumuzdan 3d perspektifte jointleri tekrar hizlamamız gerekecek çünkü tüm joint'ler 3d grid' in üzerinde oluşturuldu şu an:



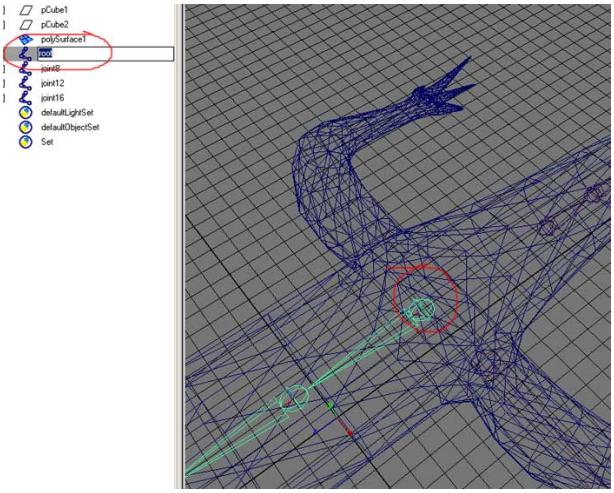
Şekil 81.

Joint'lerin başlangıç noktalarından taşıyarak geometrimde uygun yerlere taşıyorum:



Şekil 82.

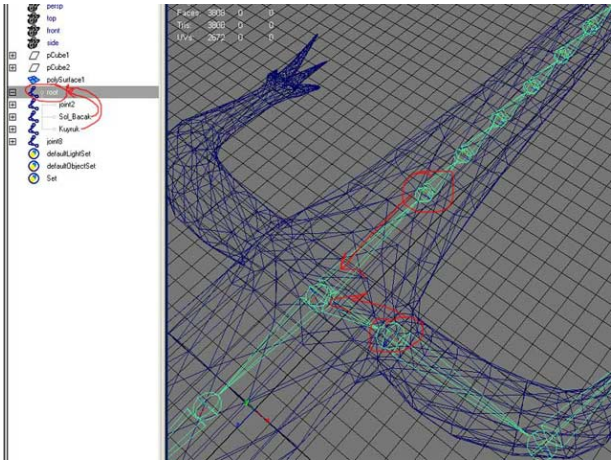
Hemen ardından TÜM JOINTLERI RENAME ediyorum. İsimlendirme çok önemli, daha sonra bu konuda başınız ağrıyabilir. Dolayısıyla önceden tüm jointlerinizi isimlendirin:



Şekil 83.

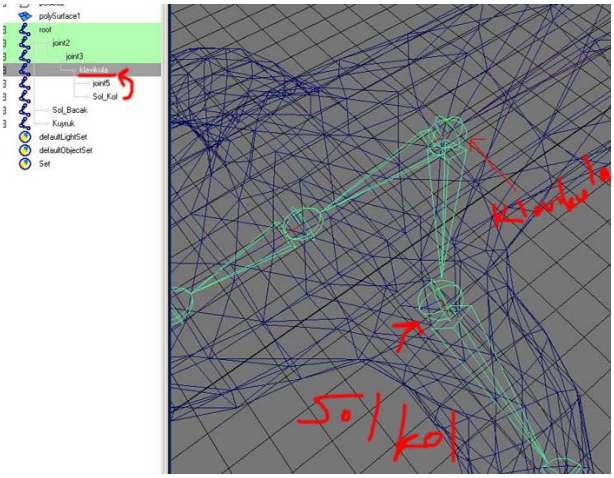
Hips joint' imiz bizim root jointimiz. Diğer tüm jointler anatomik olarak neyin karşılığı ise öyle isimlendirin, kollar bacaklar, ayaklar vesaire.

Daha sonra oluşturduğumuz tüm joint'leri root joint' e parent'liyoruz, yani Outliner' da orta mouse tuşuyla jointleri seçip root jointimizin üzerine taşıyoruz:



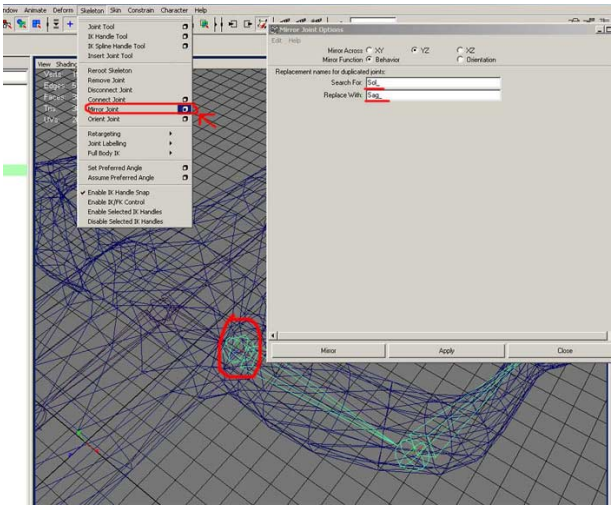
Şekil 84.

Kolların birleşeceği joint anatomi de "collar bone" olarak geçer, Türkçe' de klavikula deniyormuş, isimlendirin ve kolu bu joint' e parent' leyin.



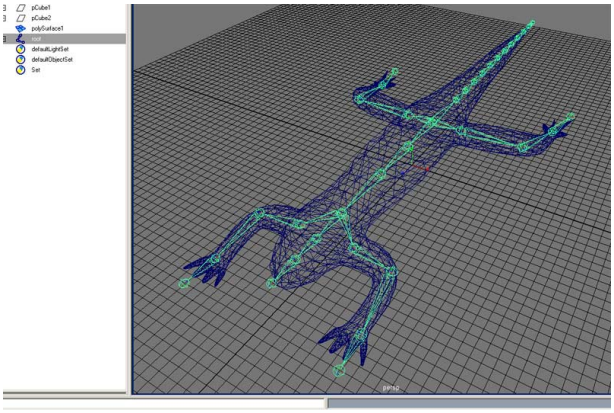
Şekil 85.

Daha sonra kolu ve bacağı mirror' layacağız. Opsiyonları aşağıdaki şekildeki gibi ayarlayın;



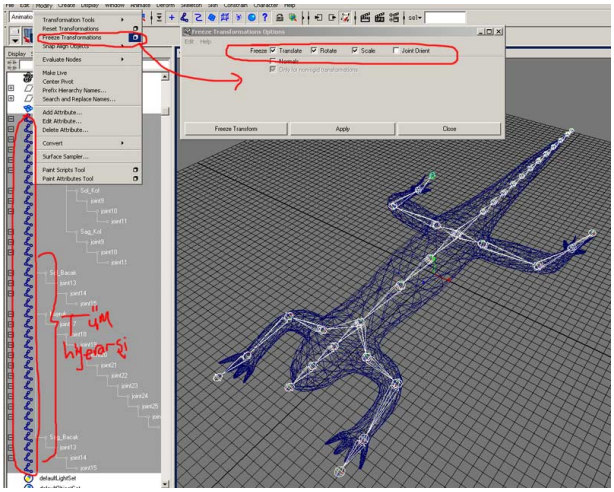
Şekil 86.

Genel görünüş şu şekilde:



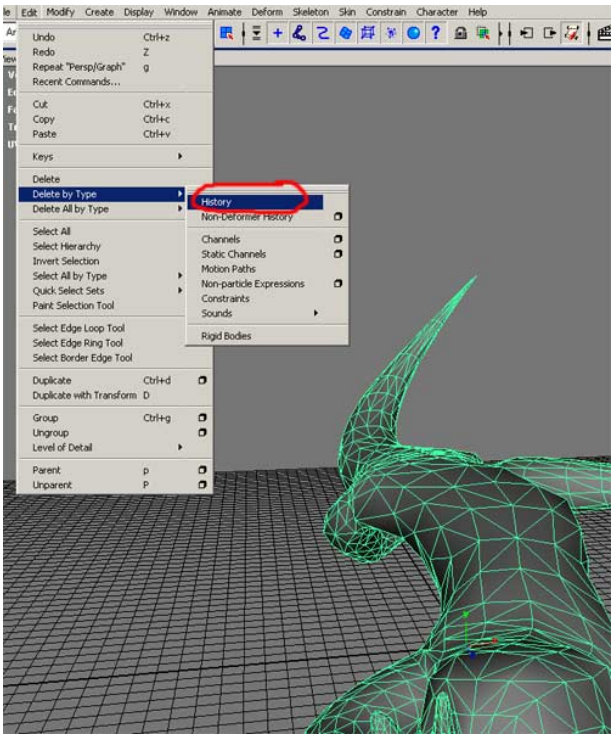
Şekil 87.

Daha sonra tüm joint' leri seçip Modify menüsünden freeze transform diyoruz. Çünkü Joint' lerin ilk rotation değerlerinin 0 olması gerekmekte. Aksi halde kertenkele geometrimiz ile joint'leri "ilişkilendirirken" (Binding) sorunlar yaşarız.



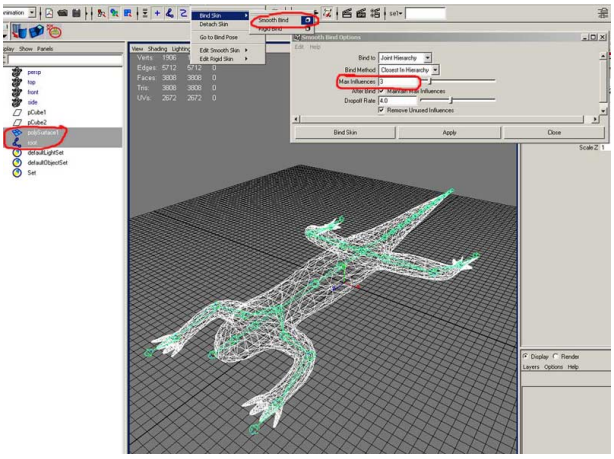
Şekil 88.

Sonra bu kez objemizi seçerek "Delete history" diyoruz. Bu da objemizi "freeze" etmek için. Yani tüm creation geçmişini siliyoruz. Bunları yapmaktaki nedenimiz sahnemizi temizlemek, diğer programlara import export' ta sorun çıkmasını da engellemek.



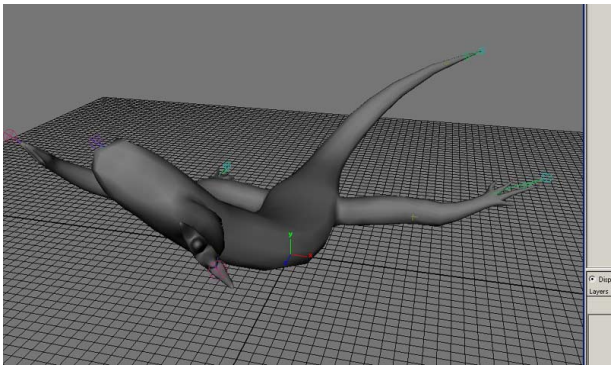
Şekil 89.

Şimdi sıra geldi iskeletimize objemizi "giydirmeye". Yani maya' nın deyimi ile binding yapacağız, hemen Skin menüsüne gidip Bind Skin> Smooth Bind diyoruz:



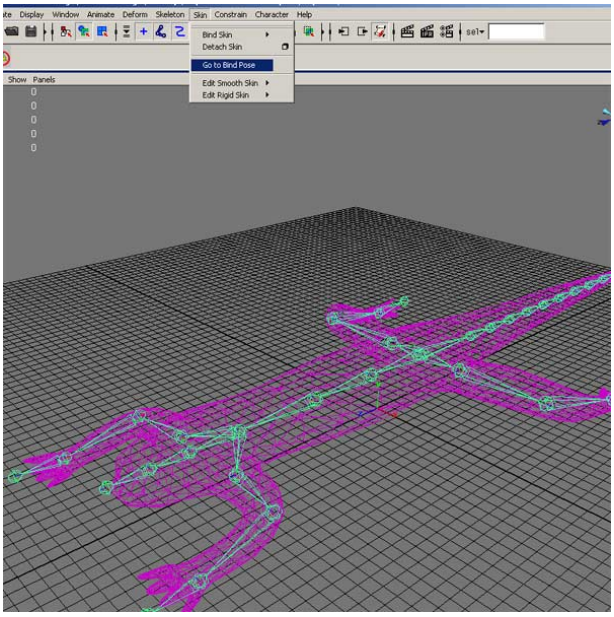
Şekil 90.

Ve iskeletimizin deformasyonunu jointleri rotate ederek test ediyoruz;



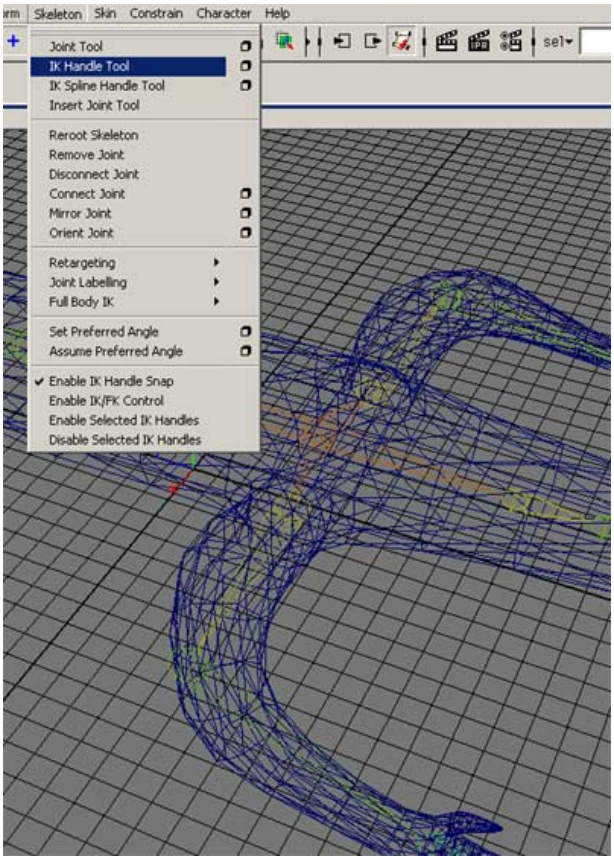
Şekil 91.

Skin> Go to bind pose, komutu ile başlangıç pozumuza geri dönebiliriz;



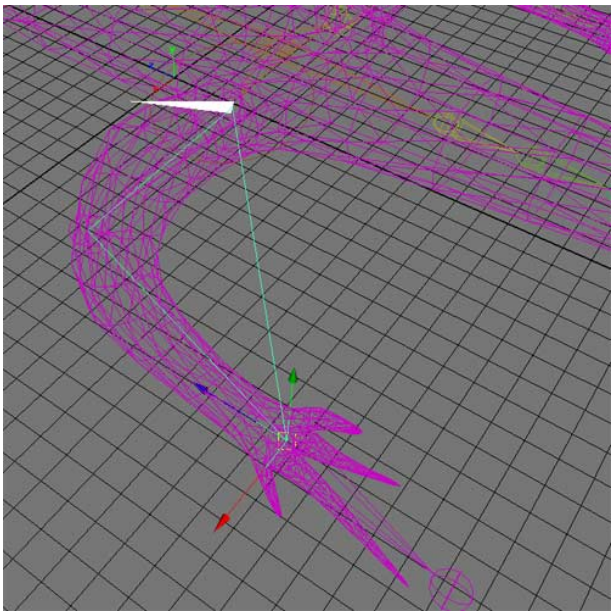
Şekil 92.

Şimdi sıra geldi Inverse Kinematics' lerimizi oluşturmaya.



Şekil 93.

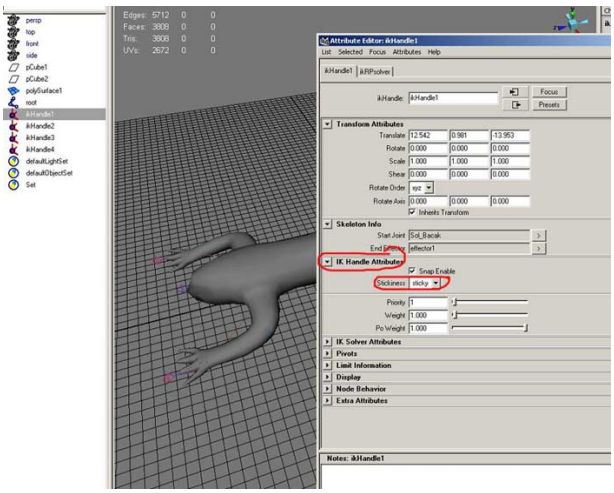
Inverse Kinematics'ler yada kısaca IK' ler kol ve bacakları animate etmede bize büyük kolaylıklar sağlayan bir metod.



Şekil 94.

Sol bacaktan başladık, sol bacağa tıklayıp, daha sonra ayağa tıkladığınızda IK otomatik olarak uygulanır. Yukarıdaki şekilde görüldüğü gibi IK' imiz hazır.

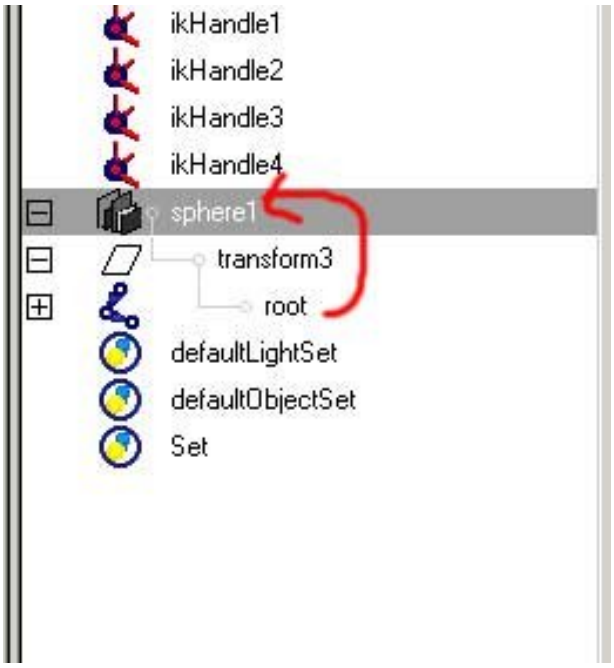
Fakat IK' imizin özelliklerinde bir değişiklik yapmamız lazım, çünkü IK' imiz "Sticky" değil, yani sabit değil. Bunu değiştirmek için IK' imizin attribute editörünü açıyoruz ve Sticky haline getiriyoruz:



Şekil 95.

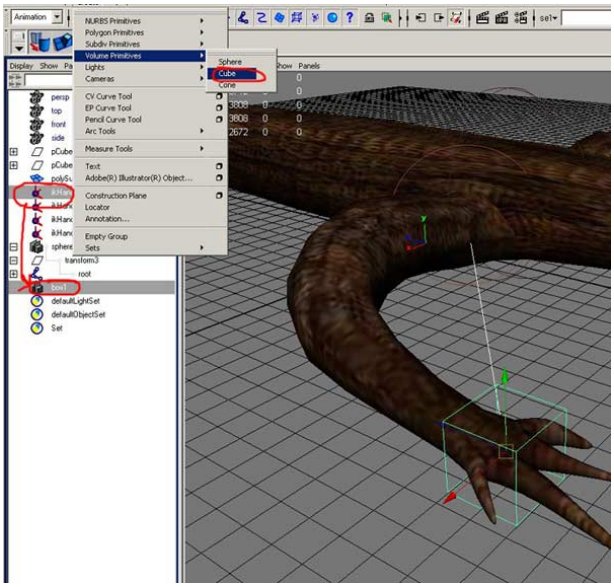
Diğer bacağa ve kollara da aynı metodla IK' lerimizi ekliyoruz.

Şimdi de 'Rigging' denilen, animasyonumuzu daha rahat yapmak için oluşturacağımız kontrol objelerimize sıra geldi. Ilk once root joint' için bir volume sphere oluşturuyorum. Root joint' i bu sphere temsil edecek artık:



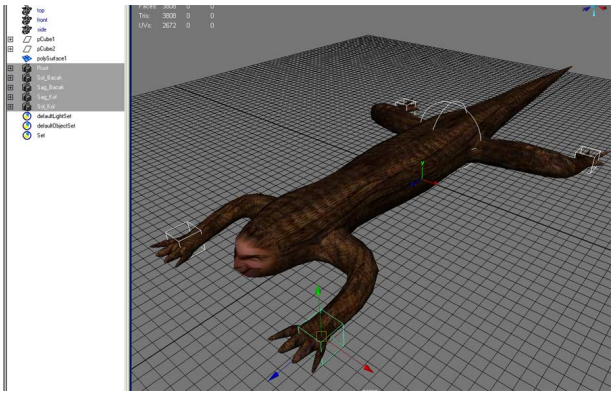
Şekil 96.

Root Joint' imi Volume sphere' ima Parent' liyorum:



Şekil 97.

Kollara ve bacaklar için volume Cube uyguluyorum, fakat bu kez sadece IK' leri parent'liyorum;

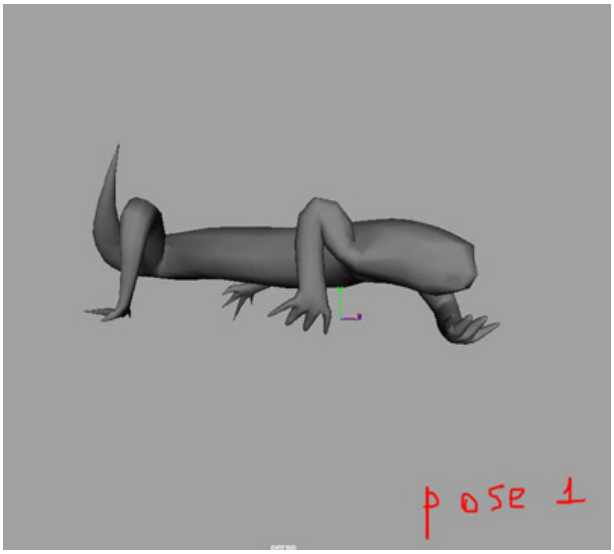


Şekil 98.

Şimdi animasyon yapmaya hazırız... Çok kısa bir animasyon yapalım, konuyu fazla uzatmayalım. 24 frame' lik bir "Walk-cycle", yürüme animasyonu yapacağız. 5 adet key imiz var:

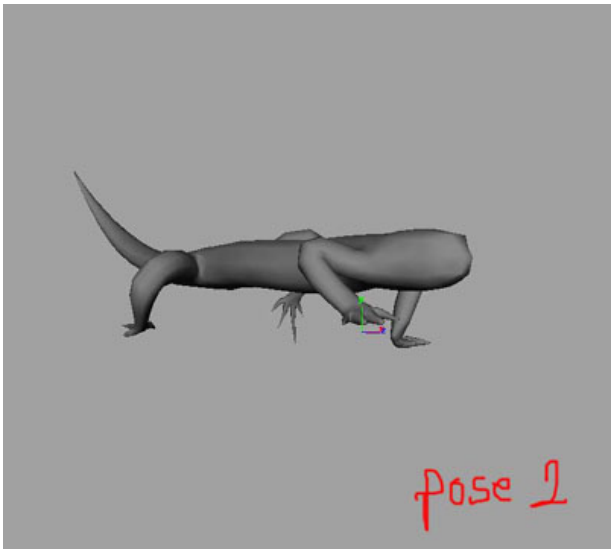
1 inci frame : Poz 1 6 ıncı frame : Poz 2 12 inci frame : Poz 3 18 inci frame : Poz 4 24 uncu frame : Poz 5 (Poz 1' in kopyası.)

1 inci frame de 1. poz için kertenkelereyi rigging nesnelerimizi kullanarak şu şekle getirmeye çalışın:



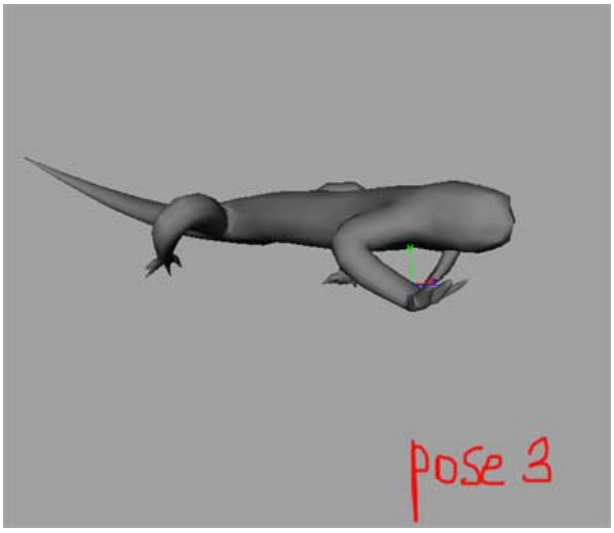
Şekil 99.

6 ıncı frame ' de ikinci poz için:



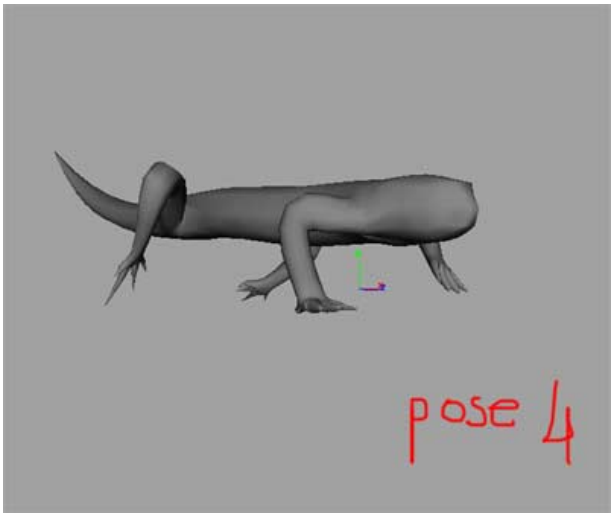
Şekil 100.

12 inci frame' de üçüncü poz:



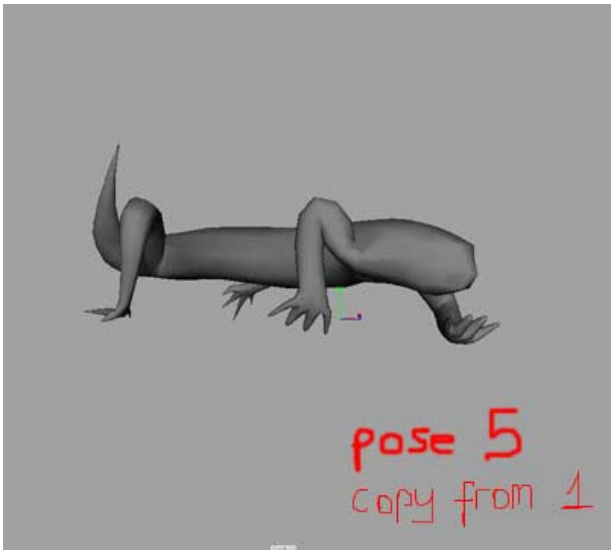
Şekil 101.

18 inci frame' de dördüncü poz:



Şekil 102.

24' üncü frame'de animasyonu Loop' a sokabilmek için ilk frame' in kopyası:

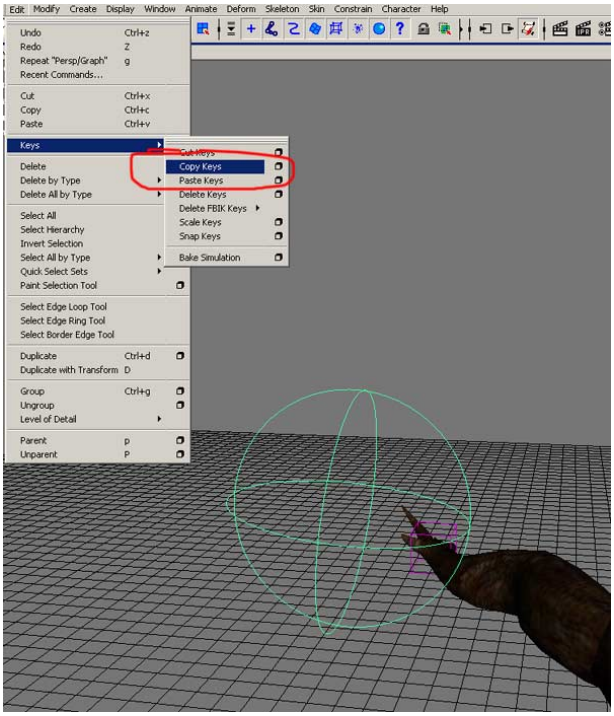


Şekil 103.

Evet animasyonumuzda bitti. Yanımdaki arkadaşa gösterdim "Olduğu kadar abi sktir et." dedi. Buradan anlıyoruz ki animasyon bi boka benzememiş, ehueh.

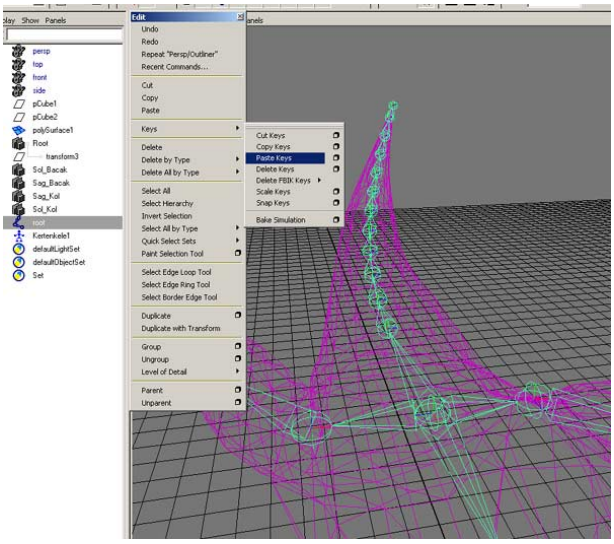
Şimdi yaptığımız animasyonu "rig" den joint' lere aktaralım ki kertenkelemizi başka programlara export ederken sorun çıkmasın.

Bunun için önce Volume Sphere 'deki animasyon data' sını root jointimize aktaralım. Çok basit, önce "unparent" ediyoruz ikisini, daha sonra Volume Sphere' i seçip Edit> Copy Keys diyoruz,



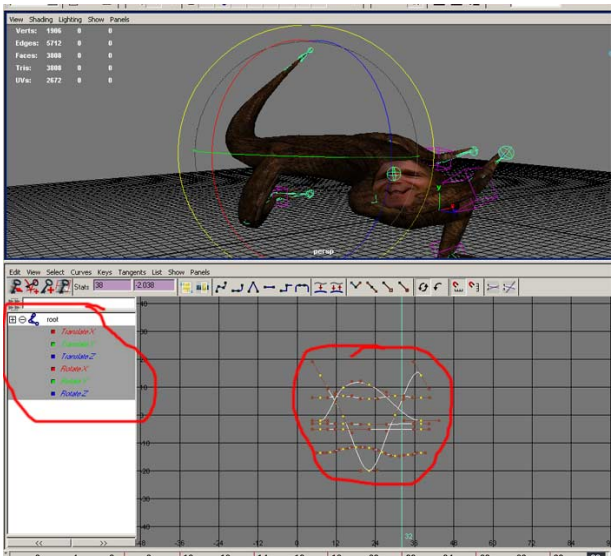
Şekil 104.

sonra root joint' imizi seçip Edit> Paste Keys diyoruz.



Şekil 105.

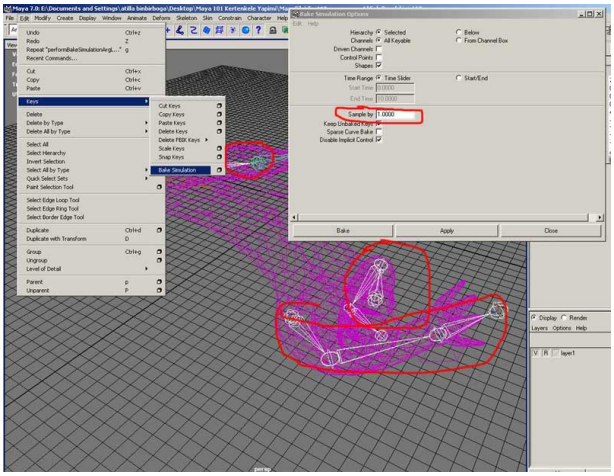
Burada animasyonun "Offset" inde hafif kaymalar olabilir, o yüzden Window> animation editors> Graph editor' de Root joint' imizin curve'lerini hafifçe yukarı yada aşağı kaydırarak orjinal yerine getiriyoruz:



Şekil 106.

Kollara ve bacaklara IK uygulamış olduğumuzdan onların animasyon datalarını Jointlere aktarmada "Bake simulation" komutunu kullanacağız. Tüm kol ve bacak JOINT' lerini seçip Keys> Bake Simulation diyoruz:

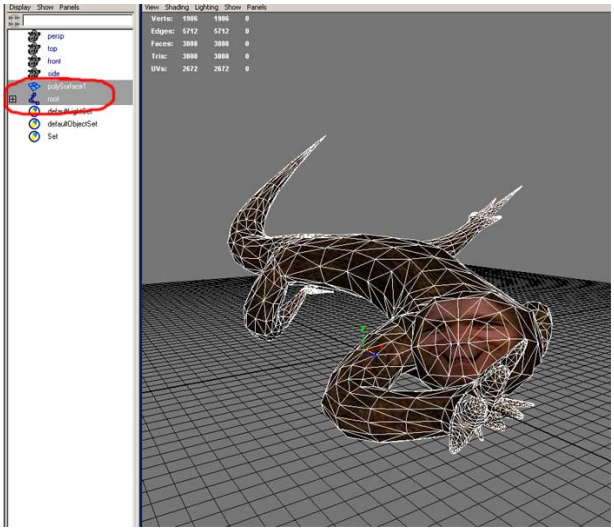
Değerler aşağıdaki gibi:



Şekil 107.

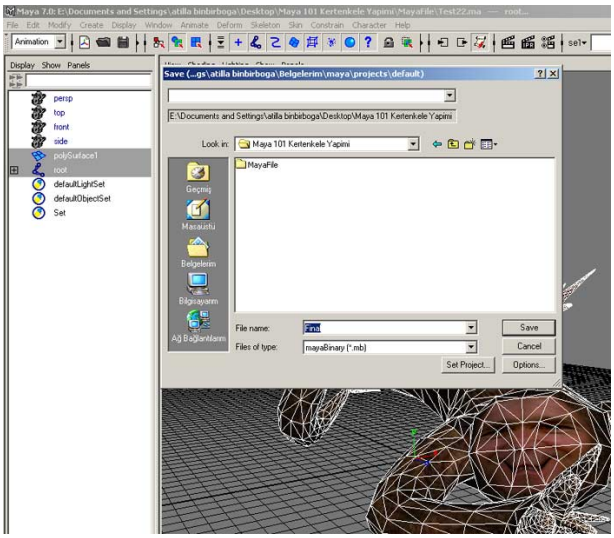
Ohh bee. Sonunda bitti. Export edebiliriz. Gelin şimdi bunu Milkshape formatına dönüştürelim ki bu animasyonu demolarımızda kullanmamız kolay olsun.

Önce maya sahne dosyamızın temiz olduğundan emin olalım. Sahnemizde Kertenkele geometrimiz ve Jointlerimiz var, başkada bir şey yok, olması da gerekmiyor:



Şekil 108.

Save as, diyoruz, maya binary ya da ascii farketmez.

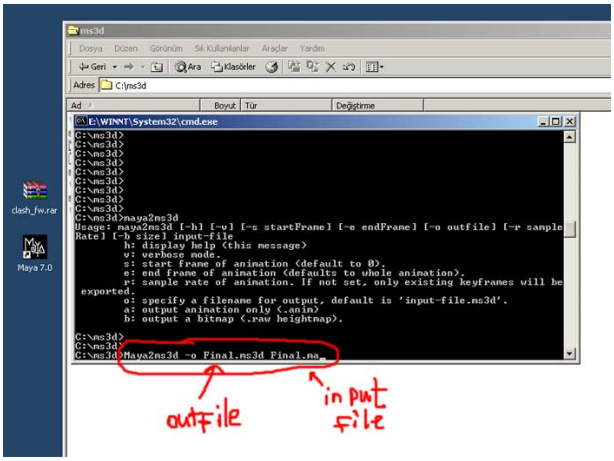


Şekil 109.

Şimdi ise internet'ten bulduğum gayet sağlam bir tool' u tanıtmak istiyorum. Google' dan maya2ms3d yazarak bulabileceğiniz bu ufak tool, direkt maya sahne dosyasını ms3d' ye çeviriyor. Muhteşem bir şey.

Readme dosyasından ayrıntılarını okuyabilirsiniz,

Aşağıda convert yaparken ki hali var:



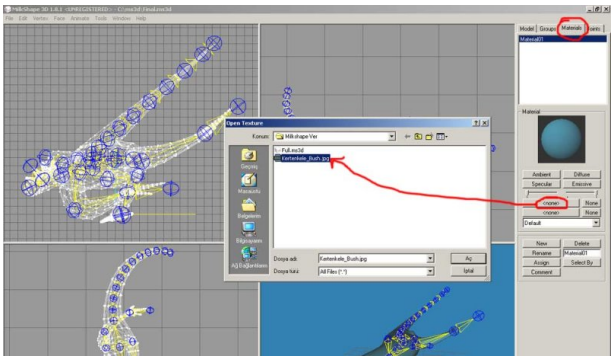
Şekil 110.

Kısaca

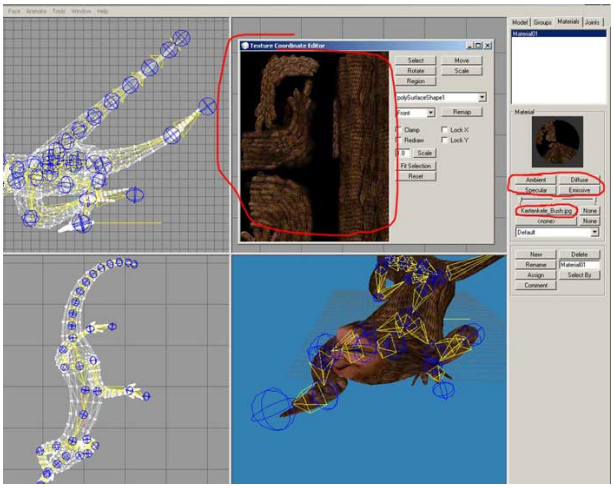
maya2ms3d -o dosyaadi.ms3d mayafileadi.ma

diyerek bitirdim. Bazi durumlarda ilk başlangıç frame' i ve son frame' i yazmakta gerekebiliyor. Diğer ayarları es geçebilirsiniz.

Sıra geldi milkshape' e. Export ederken UVlayout' umuzu da export ediyoruz tabii ki. Fakat texture' ümüzü Milkshape' de bir daha atamamız gerekiyor:



Şekil 111.



Şekil 112.

Gördüğünüz gibi çok kısa bir süre içerisinde kertenkelemizi modelledik texture' ledik ve animasyonunu yapıp ms3d formatında çıkardık.

Okuyun araştırın bu aşamaların ayrıntılarını öğrenin siz daha da güzellerini yapın. Ayrıca bu tutorial' la birlikte kertenkelenin ms3d dosyasını ve Quicktime' da animasyon loop' unu da koydum, (Ed: BonusDisk'te bulabilirsiniz) onlara da bir göz atarsınız.

İşte böyle. Haydi hoşçakalın.

Drey'in Sequencer Köşesi

Emir 'Drey' Diril

Giriş

Tüm scenerlara ve scener olmak isteyenlere merhaba. Müzik, demoların vazgeçilmez bir parçası, ve aklınızdaki müziğin hayat bulabilmesi için aracı yazılım ve donanımlara da çok iyi hakim olmak gerek. Türk demoscene topluluğunda bulunduğum kısa süre içerisinde gördüğüm kadarıyla retro-donanımların (c64, amiga, vs..) revaçta olmasından dolayı "tracker" temelli müzik yapımı hakim. Bu konuda, Hydrogen, Wisdom ya da Impetigo gibi ustalar varken bana yorum getirmek düşmez, zaten hiç niyetli de değilim :D Benim bu noktadaki hedefim, demoscene' e ve bilgisayar destekli müziğe adım atmak isteyen arkadaşlara, basit seviyeden başlayarak "sequencer" ları tanıtmak, ve bunları kullanarak kendi müziklerini yaratmaya başlamaları için bir temel oluşturmak. Haydi hayırlısı :D

Öncelikle sorulması gereken soru; sequencer nedir? Sequencer, ölçeklendirilmiş belli bir zaman çizelgesine göre kaydedilmiş ses, nota, akor, ritim gibi müzikal öğeleri dijital olarak saklayan ve bunları gerektiğinde başka bir elektronik müzik aletine aktaran yazılım ya da donanımlardır. En büyük özellikleri ve güzellikleri hem MIDI hem de AUDIO' yu aynı ortamda kullanabilmemize olanak sağlamalarıdır. Günümüzdeki en yaygın sequencer örnekleri Cakewalk Sonar, Steinberg Cubase, Propellarhead Reason, MOTU Digital Performer, Apple Logic Studio ve DigiDesign Protools' dur. Peki sequencerlar ile trackerlar arasındaki (birinin yatay, ötekinin dikey ilerlemesi dışında :P) fark nedir? Hmm, aslında bu konuya herkesin farklı bir cevabı olabilir tabi. Benim şahsi görüşüm sequencerların trackerlara göre daha kullanıcı dostu olduğu yönünde. Bu da sanırım, sequencerların, trackerların bir sonraki nesli olmasından kaynaklanıyor. Ya da ben sequencerlarla daha çok vakit geçirdiğim için öyle olduğunu düşünüyorum :D

Şimdi, sequencerları daha somut olarak tanımaya Propellarhead firmasının Reason 3 adlı yazılımını incelemeye geçerek başlayabiliriz :)

REASON' ın temelleri

Reason' ı diğer sequencerlardan ayıran iki temel özellik vardır. Bunlardan birincisi ve en önemlisi, programın içinde herhangi bir ses kayıt özelliğinin olmamasıdır. Yani bir çok sequencer' ın aksine Reason' da sadece dahili olarak bulunan software enstrümanları ve pluginleri kullanabilirsiniz. Peki bu bir kısıtlama mı? Tabii ki hayır. Çünkü Reason, "rewire" özelliği sayesinde, yukarıda adını saydığım birçok "host application" ın altında "slave" modunda çalışabilmektedir. Yani bu demektir ki siz Reason' da bir müzik hazırladıktan sonra, Reason' ı Cubase altında çalıştırıp, ses kayıtlarını da Cubase aracılığıyla yaparak amacınıza ulaşabilirsiniz. Peki bu sınırlama neden var? Sanırım bu "muhafazakar" tutum, programın çok stabil ve sorunsuz çalışmasındaki en önemli etken;)

Ikinci temel özellik de, ki bu bence müzikle uğraşan biri için gayet öğretici bir özellik, program dahilinde kullanılan tüm enstrüman ve pluginlerin bir rack sistemi halinde tasarlanmış olması. Bu sayede tüm bu birimlerin gerçekte de birbirine nasıl bağlandığı hakkında görsel bir fikre sahip ölabiliyorsunuz. Özellikle mikser bağlantıları konusunda.

Zaten bu iki özelliğin neler getirdiğini ve nasıl kullanılabileciğini, tutorialimiz ilerledikçe daha ayrıntılı bir şekilde göreceğiz.

REASON ve kullanıcı arabirimi

Reason' ı ilk açtığımız zaman karşımıza tek pencere içerisine sığdırılmış olan rack sistemi ve sequencer penceresi çıkar. Ilk iş olarak, daha rahat çalışmak adına bu pencerenin sağ üst tarafındaki "yanında siyah üçgen bulunan üst üste iki dikdörtgen" butonuna tıklayıp bu pencereleri birbirinden ayırabıliriz. Böylece esas müzik yazma işlemini gerçekleştireceğimiz sequencer penceresi ve sanal donanım ayarlarımızı yapacağımız rack penceresi olmak üzere iki farklı penceremiz olur.

Şekil 1. Reason'da ayrılmış pencereler

Rack penceresinden incelemeye başlayacak olursak, en tepede Reason' ın donanım arabirimi (hardware interface) bulunmaktadır. Bu kutunun hemen sol alt köşesindeki AUDIO OUT yazısının altında iki ışık vardır. Eğer Audio card yanıyorsa bu Reason' un tek başına çalıştığını ve hangi sistemdeki hangi ses kartını kullandığını gösterir. Eğer Rewire yanıyorsa o zaman Reason başka bir program altında (Cubase, Logic gibi) çalışıyor demektir. Işte rackimizi bu temel kutunun altına kurmaya başlarız. Mesela halen bu kutunun altına kurulmuş bir sistem yoksa, kutunun hemen soluna sağ tıklayarak, gelen menüden "create" e girer ve orda rack' e dahil edebileceğimiz enstrümanlardan "mixer14:2" yi seçeriz ve böylece sistemimizin temelini oluşturacak olan mixerimizi ses arabirimimize bağlamış oluruz. Rack' e yeni birim eklemek konusunda da işin bundan sonrası hemen

hemen aynıdır zaten. Sadece birşey eklemek istediğimiz birime sağ tıklayarak gelen menüden create'e girip bu birime eklemek istediğimiz birimi seçiyoruz. Bu birim mixere ekleyeceğimiz yeni bir enstrüman ya da bir enstrümana eklemek istediğimzi bir efekt olabilir. Eğer bir enstrüman eklerseniz, o anki mikserde boş olan ilk kanala atanacaktır. Eğer bir efekt eklerseniz, o anki mikserde boş olan ilk aux kanalına atanacaktır (bunların sırasını sağ üstteki "return" yazan kutucuğun altında görebilirsiniz). Eklenen efektlerin herhangi bir enstrüman üzerindeki etkisini, ilgili enstrümanin kanalında en üstte bulunan aux knobları ile yapabilirsiniz. Örnek olarak Mixerimize reverb eklediğimizi ve bunun da aux 1' e atandığını düşünelim. Bu durumda herhangi bir kanaldaki aux 1 knobunu çevirerek o kanaldaki reverb miktarını ayarlayabiliriz. Mixer ile ilgili daha fazla detaya, ileride her bir birimin ayrıca incelemesini yaparken gireceğiz.

Gelelim Sequencer penceresine. Burada müziğinizi oluştururken ihtiyaç duyacağınız tüm butonlar yukarıya dizilmiş vaziyettedir. En soldaki kutucuk ile bu pencerenin iki temel işlevi olan düzenleme (arrange) ve yazma-düzeltme (edit) modları arasında geçiş yaparız. Bu butona tıklayıp edit moduna geçtiğimizde, hemen yanındaki ayraçla ayrılmış olan beş buton aktif hale gelir. Bu beş butondan ilki piano roll'u gösterir (piano roll, tüm sequencerlarda notaların kaydedilmesi için kullanılan temel sistemdir. Soldan sağa doğru zaman zaman ilerler ve nota süresi belirlenir, dikey doğrultuda da kullanilacak nota seçilir, yukari çıkıldıkça nota tizleşir) Hemen yanındaki merdiven resimli buton Reason' ın loop temelli enstrümani olan Dr. Rex içindir. Piano roll' daki gibi notalar yerine REX looplarının dilimlerini gösterir. Bir yandaki ikon, trampet resminden de anlaşılacağı üzere Redrummer' a özel drum roll' dur. Yine piano roll' daki gibi notalar yerine Redrum' da seçilen patch' e göre davulun elemanlarını gösterir (kick, snare, hihat. vb). Bundan sonraki ikon, bu ilk üç görünümden herhangi birinde girilen notaların özelliklerini göstermeye yöneliktir (velocity gibi). Son iki ikon ise genel olarak o kanaldaki enstrümana bağlı bazı otomasyonları gösterir ve düzenlenmesini sağlar. Zaten altıncı ikon olan mavi ikona basılınca hangi otomasyonu göstereceği de bu ikonun hemen yaninda bulunan ikona tıklayıp gelen dropdown menüden seçilir. Neyse daha fazla karmaşaya girmeden esas editing toollarına geçelim.

Bu ikonlardan sonra başlayan altılı bir ikon seti daha vardır. Işte biz bu altılı seti kullanarak istediğimiz notaları sisteme gireriz ya da düzeltiriz. En soldaki ok ile bize herhangi bir notayı ya da nota grubunu seçme şansını verir. Onun yanındaki kalem ile notalarımızı yazarız. Bir sonraki silgi ile anlaşılacağı üzere yazılı olanları sileriz. Sonraki çizgi çekme tooludur. Mesela bazı otomasyonları kontrol ederken kalem tool' u ile yeterince düzgün çizgi oluşturamayacağınız için bu düz çizgi tool'unu kullanmanız gerekebilir. Bunun yanındaki mercek ile görüntüyü büyütürüz. En son el tool'u ile de pencerenin içini tutup sürükleyerek piano roll' da gitmek istediğimiz yere gidebiliriz.

Altılı ikon setini de geçtikten sonra geriye nota bazında çözünürlüğü ayarladığımız menü kalır. Burada genel olarak 1/16 seçilidir. Yani zaman çizelgesi bir vuruşun 16 da 1'i olan küçük parçaçıklara bölünmüştür. Bundan sonra gelen mıknatıs ikonu, "snap to grid" özelliğini aktif hale getirir. Yani notaları o anki çözünürlük birimine bitişik olarak yazmanızı sağlar, böylece nota değerlerinde hata yapmazsınız. Pencerenin bu üst barındaki son bölüm de "quantizing" bölümüdür.(Notaların quantize edilmesi, belli bir nota ya da nota grubu seçildikten sonra önceden belirlenmiş bir ölçeğe göre zaman çizelgesinde hizalanmasıdır. Yani 16'lık notalardan oluşan bir partiyi midi klavyede çalarak kaydettiniz. Ama notalar tam olarak grid' e denk gelmediği için yer yer ritim dışı (off beat) duyuluyor. Işte bu noktada quantize özelliğinden faydalanarak bu ritim dışı notaları grid' e gelecek şekilde hizalatıp tekrar ritme uygun hale getirebiliriz.) Quantize bölümündeki ilk ikonumuz notaların kayıt sırasında daha çalınırken hizalanmasini sağlar. Hemen yanında yine bunun çözünürlüğünü drop-down menüden seçebiliriz. Bir sonraki ikon ise zaten keydedilmiş ya da yazılmış olan notaların hizalanması içindir. Son olarak en sağdaki yüzdelik kutusu ile de bu "quantize" işleminin hangi hassasiyette yapılacağını seçmek içindir. Eğer 100 % seçili olursa bu durumda notalar tam olarak grid çizgisinin üstüne getirilecektir.

Bu noktada ufak ufak birşeyler üretebilmeye başlamamız adına açıklama gerektiren bir bölüm daha var o da her iki pencerenin de en altında yer alan kontrol paneli. Bu panelde de yine bir sürü ayar bulunmakla beraber şu an için bilmemiz gereken üç kutu var. Bunlardan birincisi ve en önemlisi ortada bulunan transport paneli. Bu panel "stop, play, forward-rewind, ve record" düğmelerini içerir. Bunun hemen altında sequencer çubuğunun o an bulunduğu pozisyon "ölçü, vurus, 16'lık nota" şeklinde dijital rakamlarla gösterilir. Bu kutunun solunda ise müziğin BPM (beat per min) temposu ve zaman anahtarı bulunur (3/4lük, 4/4lük, 9/8lik gibi). Hemen solunda ise Metronom bulunur. Click butonundan metronomu aktif hale getirebilir ve altındaki knobdan ses seviyesini ayarlayabiliriz. (Bu bölümde yer alan diğer kısımlar daha sonra açıklanacaktır)

Hızlı başlangıç

Birimleri daha detaylı olarak görmeye başlamadan önce, hemen müzik üretimine başlamak için elimizin altında neler olduğuna bir bakalim. Öncelikli olarak Reason' da kullanabileceğimiz iki tip mikser bulunur. Bunların birisi 14, diğeri 6 kanal olmak üzere ikisi de stereo çıkış sahibidir. Bunun dışında davul partilerini yazabilmemiz için Redrum adlı bir drummachine vardır. Bunun dışında Dr.Rex adında küçük parçalara ayrılmış looplar kullanan bir loop player bulunur. Son olarak da SubTractor ile Malstorm adlarında iki adet synthesizer ve NN19 ile NNXT adında iki adet de sampler bulunur. Bu enstrümanların yanı sıra Reason' da bir de Combinator adı verilen bir birim mevcuttur ki, bir enstrüman ağacını olduğu gibi tüm efekt ve uzantıları ile kaydedip sonradan tekrar kullanabilmemizi sağlar. Mesela diyelim ki bir kaç enstrüman ve efekt zincirinden oluşan çok hoş bir ton yakaldınız ve bunu tekrar tekrar başka projelerde de kullanmak istiyorsunuz. Bu durumda yapacak olduğunuz, bu zincirin başinda bulunan enstrümani seçip sağ tıklayarak gelen menüden "combine" ı seçmektir. Böylece hepsi bir combinator patch' i haline gelir ve bu şekilde bir patch olarak kaydedilebilir.

Reasondaki tüm sanal enstrüman ve efektlere patch yüklemek ve kaydetmek aynı şekilde gerçeklesir. Öncelikle her enstrümanın üstünde bulunan üçlü buton grubunu bulmalısınız. Örnek olarak SubTractor' da hemen sol üstte bulununan bu grupta sırasıyla "yukarı-aşağı ok, dosya, disket" ikonları bulunur. En soldaki o an seçili olan patch dosyası içindeki patchlerin arasında çabuk seçim için, ortadaki patch yüklemek için, en sağdaki de o

anki patch' i kaydetmek içindir.Bu buton grubunun sırası ve şekli bütün Reason enstrüman ve efektlerinde aynıdır.

Tüm bu sanal enstrümanlar için Reason dahili olarak bir sürü patch ile beraber gelir. Bunun dışında kendi patchlerinizi de oluşturup kaydedebilir ve internetten başkalarının yarattığı patchleri de indirip kullanabilirsiniz.

Sonuç

Umarım buraya kadar anlattıklarımla sequencer kavramına ve Reason' a hızlı bir giriş yaparsınız. Bu köşenin devam eden yazılarında Reason' ı ve birimlerini daha detaylı olarak tanıtmaya devam edeceğim. Hepinizin ürettiği güzel müzikleri Demo Partilerinde duymak üzere, müziksiz kalmayın ;)

Bir SID'in hikayesi

Hüseyin 'Wisdom' Kılıç

(Editörün Notu: Bu yazı ilk olarak Wisdom tarafından tr-demoscene forumuna yazılmıştı. Yazının çok değerli bilgiler içerdiğini düşündüğümüz için, kendisinin de onayını alarak Plazma'da yayınlıyoruz. Yazı ile ilgili sorularınız olursa, tr-demoscene forumunda sorabilirsiniz.)

Selamlar,

Yine Hindistan'dayım. İki gece önce ofisteyken aniden SID yapasım geldi ve hemen birşeyler yaptım. Dün Vigo ve Hydrogen'e dinletmiştim, sağolsunlar güzel şeyler söylediler.

Bügün aklıma bir fikir geldi: Henüz bu SID'i yeni yapmaya başladığıma göre, bir müziğin yapım aşamalarını merak eden insanlarla paylaşabilirim. Böylece bir müzik nasıl oluşuyor insanların daha iyi bir fikri olur. İlginç gelir mi bilmiyorum. Ama ilk üç aşamayı iletiyorum. Maalesef müziği yapmaya başlarken aklımda böyle bir fikir olmadığından çok sık kaydetmemiştim aşamaları. (Notebook'ta C64 emülatör ile çalışınca "aman yaptığım şey uçmasın" kaygısı olmuyor haliyle.)

Her bir aşama ilgili biraz detay vermeye çalışacağım, sorular olursa onları da yanıtlamaya çalışacağım, ancak bu thread'i bir SID tutorial'ına dönüştürmeye şimdilik vaktim yok maalesef.

Dosyalar iki parça halinde olacak:

1. corrupt??.dat
2. corrupt??.sid

.sid dosyasını SIDPlayer (ya da SID plugin'i destekleyen herhangi başka bir player; XMPlay, WinAMP vb.) ile açabilirsiniz.

.dat dosyası aslında rename edilmiş bir .prg dosyası. Bu dosyayı CSDb'den temin edebileceğiniz JCH Editor V3.04 (NewPlayer V20.G4) ile C64'te açabilir ve müziğin içeriğine ulaşabilirsiniz.

Her ne kadar söylemem gerekmese de bu SID'in copyright'ı hala bana ait. :-) Henüz release edilmediğinden dolayı bir yerde kullanmak ya da içinden bölümler almak istemeniz durumunda önce bana bir sormanızı rica ediyorum. (Geçmişte bu konuda dilim yandığından bunu yazmak zorunda kaldım, kusura bakmayın.)

Sözü çok uzattım. İlk üç aşamayı aşağıdaki link'ten indirebilirsiniz (Ed: Aynı zamanda bu dosya derginin bonus diskinde de var):

http://kilic.org/download/corrupt01-03.zip

Dosyayı indirip açtıysanız her bir aşamaya tek tek bakabiliriz:

corrupt01.dat / .sid

Müziğe başlamadan önce aklımda olan grand/deep bir sound yaratmaktı. Editör'ü açtım ve basit bir bass enstrümanı yaptım. Bu ses ile klavyeden birşeyler çaldım ve bunu bass ritmi olarak yazdım bir sequence'e. Bass ritmi ortaya çıkınca müzik biraz şekillenmeye başladı kafamda.

Aynı nota dizisini kullanarak bir synth arpeggio'su yazdım. Sonra bu arpeggio'yu ikinci bir kanalda, notaları biraz aşağıya kaydırarak kullandım, böylece basit bir echo efekti oldu.

Bass sesini biraz kısık tuttum. Bu sayede daha sonra aynı ritmi yüksek sesle çaldığımda etkisi daha fazla olsun istedim. Dolgunluğu yakalayabilmek için önce zayıflığı aktarmam gerek diye düşündüm. Bu ritmi müziğin genel teması olarak kullanmayı düşünüyorum ve müziğin içinde arada bir bu ritme geri döneceğim.

Müzikte fazla tekrar olsun istemiyorum ve progression hızlı olsun istiyorum. O yüzden hafiften yankılanan basit bir ritim daha yazdım ve bunu ilk arppegio'ları çaldıktan hemen sonra çalmaya başladım. Müziğin tonu yavaş olmasına rağmen bu biraz hızlıca ritim sayesinde bir denge yakalamaya çalıştım. Bu tip kontrastları daha sonra da kullanacağım.

Enstrümanların programlanması, klavyeden birşeyler çalınması, ilk sequence'lerin yazılması için şu ana kadar geçen toplam süre: 01:48

corrupt02.dat / .sid

Bass ritminin arasına hi-hat'ler serpiştirdim. Bass ritmin akışı yüzünden bu hi-hat'ler istediğim ritmi yakalayamiyor, o yüzden şimdilik hoşuma gitmese de öyle bıraktım. Daha sonra buna tekrar döneceğim muhtemelen.

Henüz geçişleri ve atakları yapmak için erken olsa da drum başlamadan önce tek bir snare ile aklımda olan atağı kısmen yapıp bıraktım. Buna da daha sonra döneceğim.

Drum'ların başladığı yerde C64'teki üç kanal sınırlamamasından dolayı sık kullanılan bir tekniği kullandım: Bass enstrümanının ilk birkaç tick'ine base drum koydum, böylece herhangi bir bass notası çaldığımda önce base drum çalıyor, hemen arkasından da bass çalıyor. Bu sesi tek başına dinlediğimizde bile kulağımız bu gecikmeyi pek algılayamıyor. O yüzden efektif bir yöntem.

2. kanalda hızlı ve kesiksiz bir ritim olduğundan o kanalı snare'ler için kullanamam. Bass ritmi bozmamak için orada da kullanamıyorum, o yüzden snare'leri 3. kanala yazmak zorunda kaldım. Bass ritmi 3. kanala aynen kopyalayıp, enstrümanı daha gitar benzeri bir ses ile değiştirdim. Aralarda da snare oldu. Snare yazmak zorunda kaldığım tick'lerdeki gitar notalarının eksikliği bass ritmi de ayni şeyi çaldığından dolayı hissedilmiyor.

Bass ritmini aynı enstrümanın daha yüksek sesli bir varyasyonuyla çalıyorum artık. Ayrıca filtre tipini de sesi daha dolgulaştıracak şekilde değiştirdim. Sesin release'i daha önce çok kısıktı, şimdi onu arttırdım, bunun da dolgunluğa faydası oldu, ayrıca biraz yankı varmış gibi oldu.

Gitar için kullandığım enstrumanın sesi biraz kısık. Basit bir

enstrüman. Tek amacı bass'ı güçlendirmek.

Şu aşamada yumuşak bir intro'dan sonra aniden dolgunlaşan ve sertleşen bir müzik parçası var elimde. İlk baştaki amacıma biraz ulaşmış durumdayım.

Eğer kafamda komple bir parça fikriyle oturmamışsam bilgisayar başına, yazdığım birkaç ritmin varyasyonlarını bir arada çalarak parça nasıl devam edebilir anlamaya çalışıyorum. Burada da öyle yaptım, aynı ritimleri farklı varyasyonlar ve sırayla çalarak müziğin devamının gelmesi için kendime fikir vermeye çalıştım.

Gitar sesini bir oktav üstten çalarak müziğin o bölümünde hafif bir coşku ile kızgınlık hissini bir arada vermeye çalıştım. Bu kısım henüz tam istediğim gibi olmadı. Muhtemelen aynı şeyi bir oktav daha yükselterek deneyeceğim daha sonra.

Varyasyonlardan biri de gitar ritmini alıp bunu ring modulation ve synchronization ile çaldırmak ve 2. kanal ile karıştırmak oldu. Biraz futuristic bir sound oldu gibi.

Bass'ın dolgun versiyonu ile intro'daki arpeggio'ları bir arada çaldım. İntro'dan daha güzel sound ediyor ama intro'yu yine de daha sessiz ve sade bırakacağım sanırım.

Birkaç varyasyon daha var, pek söze değer değiller, onları geçiyorum.

Süre: 03:47

corrupt03.dat / .sid

En son olarak çift gitar ve bass bir arada çalıyormuş gibi denedim. Bu bölüm şu anda oldukça amatör sound ediyor, daha sonra düzelteceğim.

Ayrıca, son save'de parçanın sonuna biraz sessizlik ekledim. Loop eden müzikler "bilgisayar müziği" hissinden kurtulamıyor kanımca. Çoğu zaman bu hoşuma gitse de bu müziği direkt loop ettirmemeyi düşünüyorum. Bu iş ayrıca düzgün bir wav/mp3 export alabilmek için gerekliydi (SIDPlay'de wav export alabilmek için export süresini girmeniz gerekiyor.)

Süre: 03:53

Bilmiyorum yazdıklarım ne kadar anlamlı ama şimdilik bu kadar. Umarım biraz olsun ilginç geliyordur ya da belki birilerine yardimci olabilecek birkaç ipucu vardır. Gelen (gelirse :-)) tepkilere göre bu yazıları biraz daha şekillendirebilirim sanıyorum.

Yazıya başlarken de belirttiğim üzere teknik detaylara (tick vb.) ve özellikle de SID ile ilgili detaylara girmiyorum. Bu konularda yeterince bilgi var internet'te. Onlar hakkında da yazarsam bu yazı bitmez gibi görünüyor, biraz da o yüzden üstünden geçiyorum. Yine de kafanıza çok takılan bir soru olursa cevaplamaya çalışayım. Buraya yazarsanız ben olmasam bile bilen baska biri cevaplar mutlaka.

Müziğe birşeyler eklediğimde tekrar yazacağım.

Şimdi yazdıklarımı tekrar okudum da çoğu cümleyi kötü dil kullanımı yüzünden ben bile anlamadım. :-) Hindistan insani yoruyor, mazur görünüz.

Sevgiler,

Amiga Kickstart Rom Dosyası

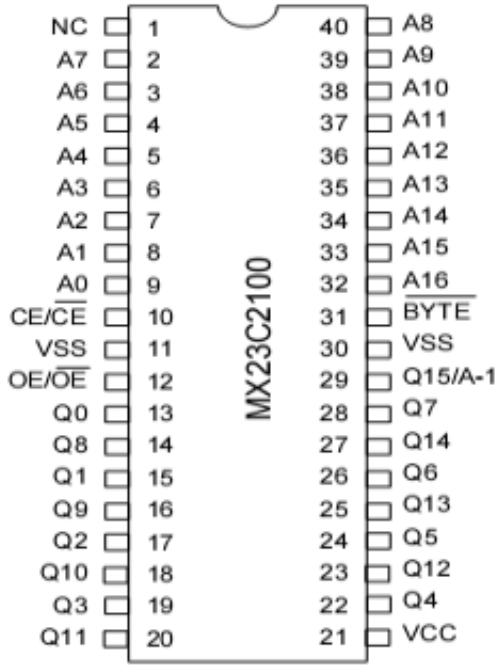
Arda 'CoZe' Karaduman

Amiga için KickStart Rom yazma

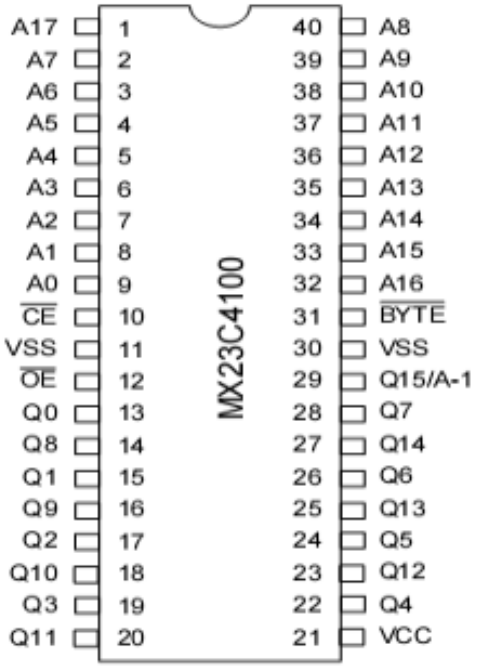
Merhaba arkadaşlar,

Bu yazımızda amigalar için kickstart rom konusunu ele alacağız. İlk bölümde hazır bir rom dosyasının nasıl gerçek çipe yazılabileceğini anlatacağım. İkinci bölüm rom dosyaları üzerinde değişiklikler yaparak kendi amaçlarınıza uygun rom dosyaları hazırlamak üzerine olacak. Üçüncü ve son bölümde ise normalde amigalarda kullanılmayan 1mb büyüklüğündeki rom dosyalarını hazırlama ve yazmayı göreceğiz.

Yazacağınız rom dosyalarını hazırlamak için amiganızdan grab edeceğiniz rom dosyası, upgrade edeceğiniz patch'ler, ve eğer güncellenmiş rom paketleri kullanacaksanız, boing bag paketlerine ihtiyacınız olacak. Rom hazırlama işlemini WinUAE üzerinde veya gerçek amiganızda gerçekleştirebilirsiniz. Bu romları gerçek çiplere yazmak için ise uygun bir rom yazıcıya ihtiyacınız olacak.



Şekil 1. 23C2100 Pinout



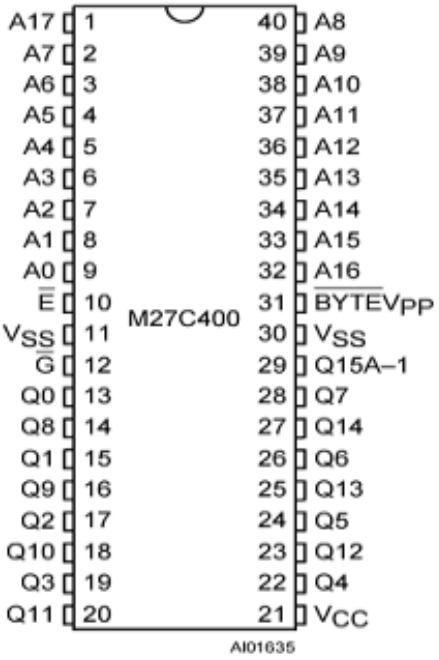
Şekil 2. 23C4100 Pinout

Öncelikle Amiga'da kullanılan rom yapısını tanıyalım. Amiga romları, kickstart 1.* serisinde 256 Kb iken, 2.0 ve üzerinde 512 Kb yer kaplar. Bu bilgi, Amiga 500, 600, 2000 modellerinde 512 Kb'lık tek bir romda toplanırken, Amiga 1200, 3000 ve 4000 modellerinde 256 Kb'lık iki roma bölüştürülmüştür. Bunun sebebi 500, 600 ve 2000 modellerinin 16 bit bus'a sahip olup kickstart'a 16 bit'lik bölümler halinde ulaşırken, 1200, 3000 ve 4000 modellerinin 32 bit bus üzerinde 32 bit'lik parçalar halinde rom'a ulaşmasıdır. Kullanılan çipler 23C serisi silinemeyen, yeniden yazılamayan romlardır. 512 Kb rom kullanan amigalarda 23C4100 serisi romlar kullanılırken, 256 Kb'lık çift rom kullanan amigalarda 23C2100 serisi romlar kullanılmıştır. Yukarıda bahsi geçen romların pinoutlarını görebilisiniz.

Yapmamız gereken, büyüklük ve pinout olarak bu çiplerle uyumluluk gösteren bir çip bulup yazmaktır. Pinout'u uymayan çipleri de gerekli sinyalleri ilgili yerlere gönderecek bir adaptör hazırlayarak kullanabilirsiniz, ama tabi yapacağınız iş artar. Tabi çipleri bulduktan sonra bu romları programlayabilecek bir eprom programlayıcıda temin etmeniz gerekiyor.

Şu anda piyasada Amiga romları ile uyumluluk gösteren romlar 27C400 ve 27C200 romlarıdır. 27C200 256 Kb olup amiga 1200, 3000, 4000 romları yerine kullanabilirsiniz. 27C400 ise 512 Kb'tır. Aslında bu romlarda oldukça eski olup bulunması gitgide zorlaşmaktadır. Özellikle 27C200 bulması çok zor oldu-

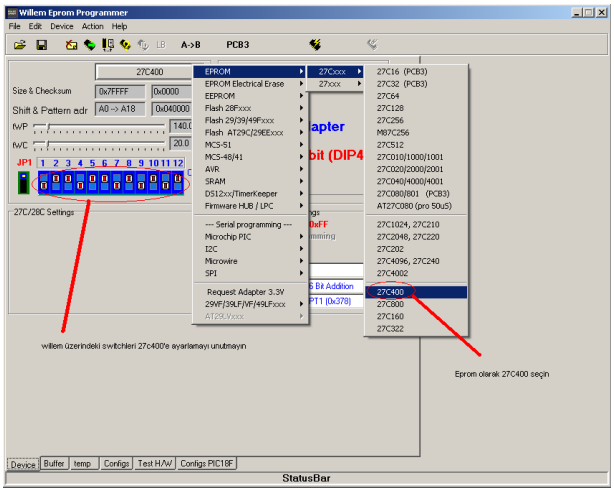
ğundan bunun yerine 27C400 kullanacağız. Pinoutlara dikkat ederseniz ikisinin arasında sadece pin 1'in farklı olduğunu görürsünüz. Bu pin 27C200'de NC (Not Connected) olduğundan kullanılmamaktadır. 27C400'lerde bu pin A17 (adres 17) pini olup, 27C400'nin üst 256 Kb kısmını adreslemede kullanılmaktadır. O yüzden 32 bit romlarda 27C200'deki datayı üst üste iki kez yazarak 27C400 çipini kullanacağız.



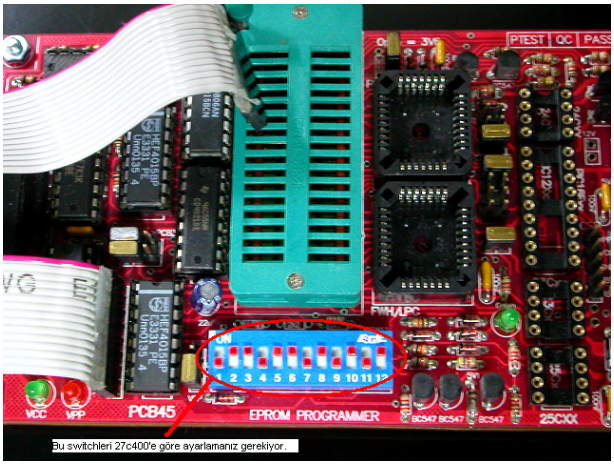
Şekil 3. 27C400 Pinout

Romlarımızı seçtiğimize göre bu romları programlayabilecek bir rom yazıcı bulmamız gerekiyor. Eskiden bu programlayıcılar oldukça pahalı aletlerdi. Ama artık 50$ civarına bu aletleri edinmemiz mümkün. Bu yazıda ele alınan programlayıcı, willem eprom programlayıcısı [http://www.sivava.com]dır. Bunun dışında 27C400 destekleyen herhangi bir programlayıcı deneyebilirsiniz. Ancak programlayıcının açıklamasını mutlaka iyi okuyun. Mesela willem, aslında 32 pin romlar düşünülerek hazırlanmış bir yazıcı olduğundan 40 pinlik 27C leri yazmak için bir adaptöre ihtiyaç duyuyor. Programlayıcınızı alırken mutlaka 27C400 yazıp yazamayacağınızı, ekstra bir donanıma ihtiyacınız olup olmadığını araştırın. Willem eprom programlayıcı almak isterseniz de mutlaka 40 pin adaptörüyle [http://www.sivava.com/buynow.php?pd=B12] birlikte alın.

Eğer programlayacağınız epromları ve eprom yazıcınızı temin ettiyseniz herşeyiniz hazır demektir. Bu noktada ihtiyacınız olabilecek tek şey, yanlışlıkla yazdığınız veya değiştirmek istediğiniz epromları silecek bir eprom silici. Kullanacağımız epromlar güneş ışığına hassas olmakla birlikte maalesef tam olarak silinmeleri güneş gören bir yerde iki üç hafta sürebiliyor. Rom yazarken siliciniz yoksa bunuda aklınızda bulundurun. Herhangi bir hata size iki haftaya veya rom silici almanıza neden olabilir.



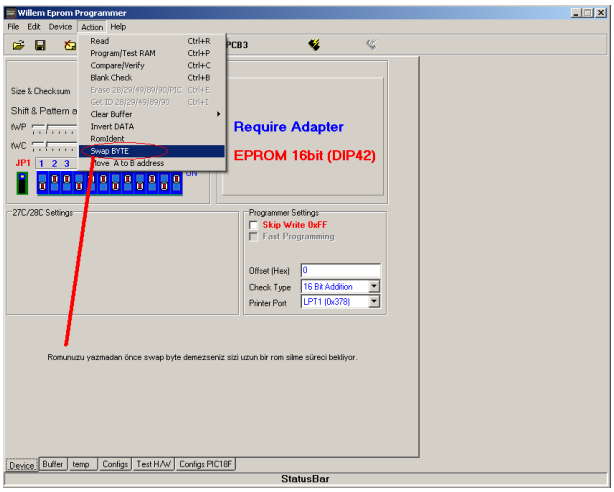
Şekil 4. Willem'de rom seçimi



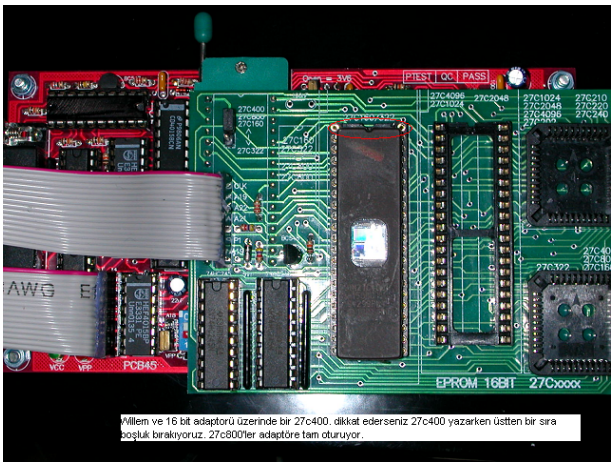
Şekil 5. Eprom yazıcı üzerindeki ayarlar

512K'lık tek parçalı romları (A500-600-2000) 27C400'lere yazmak oldukça basit bir işlem. Öncelikle rom yazıcınızı yazacağınız rom'a göre ayarlamanız gerekiyor. Willem programlayıcı da yazacağınız rom olarak 27C400'ü seçtiğinizde ekrana gelen görüntüden programlayıcı üzerindeki switchleri düzgün bir şekilde ayarlayın. Çipi adaptöre takıp adaptörü ZIF sokete yerleştirin. 27C400, 40 pin olduğundan Willem'in 42 pin adaptöründe en üstteki sırayı boş bırakmanız gerekiyor. Bundan sonra romuzun tamamen boş olup olmadığını test etmenizi tavsiye ederim (blank test). Bu işlem aynı zamanda rom yazıcınızı doğru olarak ayarlayıp ayarlamadığınızı da gösterecektir. Bundan sonra yazmak istediğiniz rom dosyasını yükleyip, bir swap byte işlemi yapmanız gerekiyor. Bu sanıyorum PC ile Amiga arasındaki endianness [http://en.wikipedia.org/wiki/Endianness] probleminden kaynaklanan bir işlem. Bunu yaptıktan sonra romunuzu yazabilirsiniz. Bir kaç dakika içinde işlemin tamamlanmış olması gerekiyor. Eğer rom tam boş değilse sorun çıkabilir. Eğer başka bir

problem çıkarsa, yazıcınızı yazacağınız roma göre tam olarak ayarlayıp ayarlamadığınızı ve yazılımda doğru rom tipini seçip seçmediğinizi kontrol etmenizi öneririm. Yazma işlemi bittikten sonra romumuzun içindeki bilgilerin silinmemesi için penceresini bir etiketle kapayıp kullanabilirsiniz !

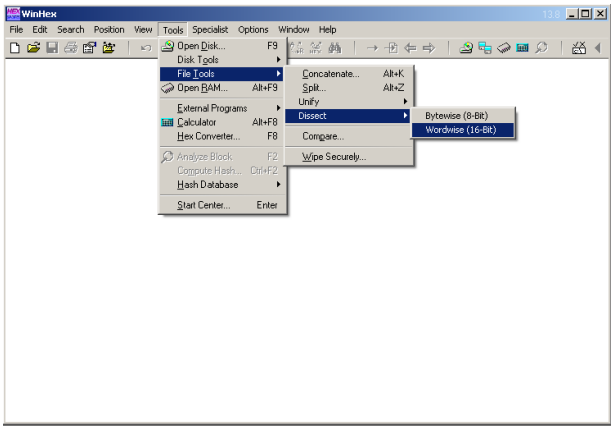


Şekil 6. Swap Byte işlemi



Şekil 7. Willem adaptörü üzerinde bir 27c400

Şimdi gelelim işin alengirli kısmına. 256K'lık iki parça halindeki romlara. Bu romları yazmadan önce iki eşit parçaya ayırmamız gerekiyor. Yanlız bu ikiye ayırma işlemi ortadan çot diye değilde, dosyanın en başından 16 bitlik bölümler halinde olması gerekiyor. Bunun sebebi bu tip romların imajını aldığınız zaman 32 bitlik wordler halinde sıralanmış olmasıdır. 32 bitlik amiga sistemlerinde bu bilgi üst 16 bit ve alt 16 bit olacak şekilde iki ayrı rom'da bulunur. Bizim ise bu dosyadan bu üst ve alt 16 bitlik rom parçacıklarını oluşturmamız gerekiyor. Bunu yapabilmek için iyi bir hex editöre ihtiyacımız var. Windows kullanıcıları için WinHex adlı editörü önerebilirim. Linux kullanıcıları zaten ne yaptıklarını biliyorlardır :)



Şekil 8. WinHex

WinHex'i açıyoruz. Yukarıda Tools menüsünden File Tools, oradan dissect, oradanda dissect wordwise (16 bit)'e tıklıyoruz. Bu işlemi yaptıktan sonra bize ikiye ayrılmış dosya için iki dosya ismi soracaktır. Burada ilkine U6A ikincisine U6B diyelim. Bu şekilde tek bir dosya halinde duran romumuzu amiga 1200'de U6A ve U6B soketlerinde kullanılabilecek şekilde iki dosyaya ayırdık. Eğer elinizde 256K romlar (27C200) varsa, bu dosyaları direk (byte swap yapmayı unutmayın) çiplere yazarak kullanabilirsiniz. Eğer elinizde 27C200 yoksa ve 512K lık 27C400'lere yazacaksanız sıradaki işlemide yapmanız gerekiyor.

Bu işlem U6A ve U6B dosyalarını arka arkaya yapıştırma işlemi. Böylece 27C400'nın boş kalan üst 256k'lık bölümünüde doldurmuş oluyoruz. Yine WinHex'de Tools menüsünden File Tools'a ve concatenate'e gelin. Size son dosyanın ismini soracaktır. İlk olarak U6A'yı oluşturalım, burada oluşturmak istediğiniz dosyaya U6A.rom ismini verin. Ondan sonra size yapıştırmak istediğiniz ilk dosyayı soracaktır. Burada bir önceki adımda yarattığımız U6A dosyasını seçin. Tekrar sorduğunda yine U6A'yı seçin. Ve done diyerek işlemi bitirin. 524288 bytes from 2 files were concatenated to "U6A.rom" mesajını gördüyseniz rom dosyamız hazır demektir. Aynı işlemi U6B içinde yapın ve U6B.rom dosyasını oluşturun.

Dosyaları oluşturduktan sonra sıra yazmaya geliyor. U6A.rom ve U6B.rom dosyalarını eprom programlayıcınızla iki ayrı 27C400'e yazın. Swap Byte yapmayı unutmayın. Yazma işlemi bittikten sonra romların üzerindeki pencereyi ışık girmeyecek şekilde etiketleyin. Artık romlarınızı amiganızda kullanabilirsiniz !

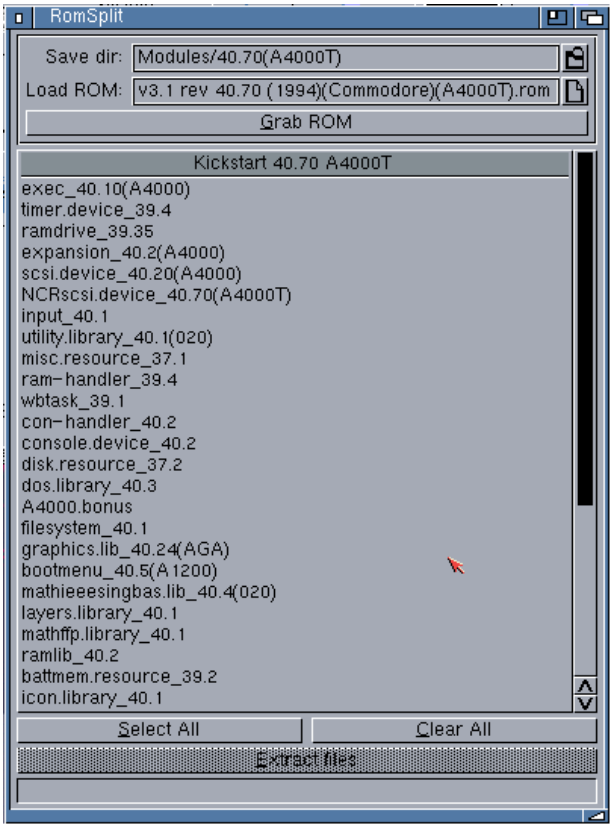
Bu arada bu işlemlerle ilgili birinci elden daha fazla bilgi almak isterseniz EAB forumundaki rom yazma başlığı [http://eab.abime.net/showpost.php?p=228868]na bir bakmanızı tavsiye ederim. Buradan faydalı bilgiler edinebilirsiniz.

Amiga için özel kickstart rom hazırlama

Yazımızın ilk bölümünde amiga için nasıl kickstart rom yazılabileceğini incelemiştik. Şimdi olayı biraz derinleştirip nasıl kendi

rom dosyalarımızı oluşturabileceğimizi inceleyelim.

Öncelikle neden böyle birşey yapmak istediğimizi açıklamakta fayda var. Maalesef bilindiği gibi amiga romları 3.1 sürümlerinde bile hatalar ve eksiklikler içermektedir. Bunları gidermek için OS 3.9 ve boing bag'ler ile bazı yamalar piyasaya sürüldü. Bu yamalar ile birlikte kickstart romlardaki bazı eksiklikler giderildiği gibi, FFS dosya sistemi, amiga 600 ve 1200 gibi onboard IDE olan sistemler için güncel IDE sürücüleri gibi güncellemeler kickstartlara eklendi. Fakat kickstart çiplerinin yeniden yazılması mümkün olmadığından bu güncellemeler software olarak gerçekleştirildi. Boing bag updateleri kurulmuş bir sistem ilk açılışta bu değişiklikleri yükleyip yeniden başlayarak kickstart'ta bulunan bilgileri güncelliyordu. Tabi bu amigamızı ilk açtığımızda gereksiz bir restart olması anlamına geliyor. Blizzard kartları ve diğer bilimum eklentiler için gereken updateler ile bilrlikte açılıştaki restart sayısı artabiliyor. Bu değişikliklerle yamalı bohça haline gelen kickstart'ı düzgün bir şekilde yeniden yazmak farz olunca, amiga sever arkadaşlarımız devreye girerek yamalanmış kickstart dosyalarını çiplere yazmanın yollarını aramaya başladılar.

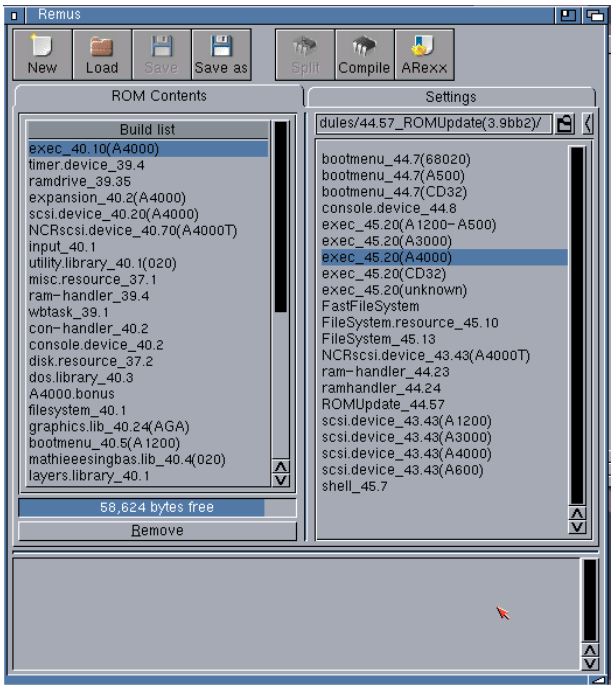


Romsplitte A4000T romunu inceliyoruz. Select all ve sonrasında extract files dersem tüm modülleri save dir bölümünde görünen dizine parçalayacak.

Grab Rom tuşu o an sistemdeki romu parçalamanızı sağlıyor.

Şekil 9. RomSplit

Öncelikle kullanacağımız programları inceleyelim. Birincisi romsplit. Romsplit programı arguman olarak verdiğimiz bir kickstart romu bileşenler haline ayırmakta kullanılıyor. Böylece biz kendimiz gerekli bileşenleri ayıklayıp işe yaramayanları atabiliyoruz, veya başka romlarda kullanabiliyoruz. Romsplit, ayrıca rom içine yazılabilir programları rom içine yazılabilir hale getiriyor. Bu ne demek derseniz, mesela boing bag paketlerindeki rom updateleri rom'a yazılabilir hale getirebilirsiniz. Aynı zamanda blizkick isimli programın bazı modulleri ve piru'nun hazırladığı exec44, romsplit yardımıyla roma yazılabilir hale getirilebiliyor.Bu programı ve remus programını amiga.org'dan doobrey'in sayfası [http://www.doobreynet.co.uk/beta/]ndan indirebilirisiniz.



Remus'ta A4000T romunu inceliyoruz. Birazdan exec.library'i boing bag 2 versiyonuyla değiştireceğim.

Şekil 10. Remus

İkinci kullanacağımız program, Remus programı. Remus, romsplit ile parçaladığımız romlardan yeni bir rom yapmamızı sağlıyor (bu arada her iki program icinde mui kurmuş olmanız ve aminet.net'ten bazı mui class'ları indirmeniz gerekiyor (BetterString, NList, urlText ve Toolbar)). Remus'un sağ penceresinde sisteminiz içinde gezinip istediğiniz parçaları sol taraftaki rom penceresine atıyorsunuz. Sisteme almak istediğiniz tüm modulleri topladığınızda compile ile romunuzu oluşturuyorsunuz. Yanlız tabi ki dikkat etmeniz gereken bazı noktalar var. Öncelikle bir rom içerisinde onca dosya arasında dosya atlamak oldukça kolay olduğundan hazır bir rom üzerinde çalışıp onun üzerinde eksiltmeler yapmanızı tavsiye ederim.

Mesela 3.1 amiga 1200 kickstart rom'unu romsplit ile parçalayın ve oluşan klasörü Remus'ta sol tarafa atın. Şu anda 1200'ün romunu tüm bileşenleri ile görüyorsunuz. Burada dikkatinizi çekebilecek birşey, exec.library'inin en üstte olmasıdır. Exec.library rom'da her zaman ilk sırada olmalıdır. Zaten remus yeni bir exec eklediğiniz zaman onu otomatik olarak ilk sıraya atar. Exec rutini

amigamızın en önemli parçasıdır. Hem bir kütüphane hemde process'ler arası mesaj alışverişi yapan bir microkerneldir.

Eğer boing bag'iniz varsa ve parçaladıysanız, boing bag rom update dizininde exec-45.20 dosyalarını göreceksiniz. Romunuzdaki eski exec'i buradaki uygun bir exec ile değiştirerek exec'inizi update edebilirisiniz. Aynı durum FileSystem, scsi.device ve diğer dosyalar içinde geçerli. Böylece OS 3.9 açılışındaki setpatch romupdate in yaptığı tüm güncellemeleri rom'a aktarabilirsiniz. Bu şekilde güncellenmiş bir rom kullanırken setpatch'in yarattiği reboot'tan kurtulmak için startup-sequence'da setpatch'i noromupdate parametresiyle çağırmanız yeterlidir. Buradaki scsi.device ve FileSystem özellikle hayati updatelerdir, harddiskinizdeki 4Gb limitini kaldırmak için bu updatelere ihtiyacımız var. Bunun yanında amiga 1200 veya 4000'iniz varsa, ve onboard ide portunu kullanmıyorsanız, açılışta ide portunun kontrol edilmesindeki bekleme sizi rahatsız ediyor olabilir. Buradan rom'unuzdaki scsi.device'ı kaldırarak amiganızın ide portunu komple devre dışı bırakabilirsiniz, böylece açılışta bekleme yaşamazsınız.

3.9 boing bag'deki tüm dosyaları kickstart'ınıza yerleştirmek istediğinizde kickstart'ınızda yeterli yer olmadığını göreceksiniz. Bu durumda kickstartınızdan bazı dosyaları çıkarabilirsiniz. Çıkarılabilecek en uygun dosyalar workbench.library ve icon.library lib'leridir. Bu kütüphaneler zaten boot sırasında libs: dizininden yüklenirler. Hatta amiga 4000T romunda workbench.library, NCRscsi.device'a yer açmak için kaldırılmıştır (amiga 4000T'de hem ide hem scsi bulunduğundan rom içinde iki scsi.device bulunur). Bu yüzden 4000T romlarındaki findWB modulü sayesinde workbench.library libs:'den yüklenir. Ama bu modul olmadan bile workbench.library yüklenebiliyor, ben pek gerek görmedim. Çıkarabileceğiniz bir diğer dosya icon.library. Bu da workbench.library gibi libs: dizininden yüklendiği için rom'da yer almasına gerek olmayan bir kütüphanedir.

Remus programı ile rom oluşturma hakkında söylenebilecekler şimdilik bu kadar, biraz kurcalayarak ve winuae ile denemeler yaparak olaya kısa sürede vakıf olabilirsiniz. Birazda blizkick ve modullerinden bahsetmek istiyorum. Blizkick programı, piru nikli bir amiga kullanıcısının geliştirdiği, blizzard ve maprom destekleyen diğer bazı turbo kartların maprom özelliğini kullanarak modifiye kickromlardan boot etmesini sağlayan bir programdır. Kullanılmak istenen kickrom fastram'a yüklenip reboot edilir ve rom çağrıları fastram'daki bu bölgeye yönlendirilir. Bu programın bir çok modulü vardır (FPU librarysindeki bir bug'ı gideren fixMath404, rom içindeki math rutinlerini optimize eden patchMath020, floppy sesini kesen NoClick gibi.). İsterseniz bunları da kickstartınıza ekleyebilirsiniz. Bu modülleri blizkick programı ile birlikte piru'nun web sitesi [http://piru.dyndns.org/~p/sw/]nden indirebilirsiniz. Yanlız bu modülleride kullanmadan önce romsplit'ten geçirmeniz gerekiyor. Bu sayfada dikkatinizi çekebilecek bir baska dosya exec44.library'sidir. Exec44, piru'nun hazırladığı, 68020+ islemciler düşünülerek yazılmış optimize, ufak ve bir çok bug fix barındıran bir kütüphanedir. İsterseniz official exec.library'inizi çıkartıp bu exec.library'ide deneyebilirsiniz. Yanlız 3.9 boing bag'lerden önce yazıldığı için 3.9 update'ler ile sorun çıkartıyor, eğer kullanacaksanız düz bir 3.1 rom üzerinde kullanmanızı tavsiye ederim. Buradan indireceğiniz modulü bir kez romsplitten geçirdikten sonra romunuza atabilirsiniz.

Bu arada 68000 işlemci kullanan Amigacı arkadaşlarımıza bir uyarıda bulunmam gerekiyor. En son Commodore Exec.library'si ve birçok rom update en az 68020 işlemci istediğinden düz amiga 500 ve 600 lerde çalışmamaktadır. Bu sistemler için en uygun romlar scsi.device ve FFS filesystem update'leri yapılmış normal 3.1 exec içeren romlardır.

Amiga için 1mb romlar

Eğer CD32, CDTV gibi konsolvari amiga sistemleri gördüyseniz mutlaka dikkatinizi çekmiştir. Bu sistemler açılışta jingle'lı, yanarlı dönerli bir boot ekranına açılırlar. Peki herhangi bir disket, CD okumadan bu konsollar bu boot ekranlarını nerden yüklüyorlar dersiniz ? İsterseniz amiga hafıza haritasına bir göz atarak bu sorumuza bir yanıt arayalım.

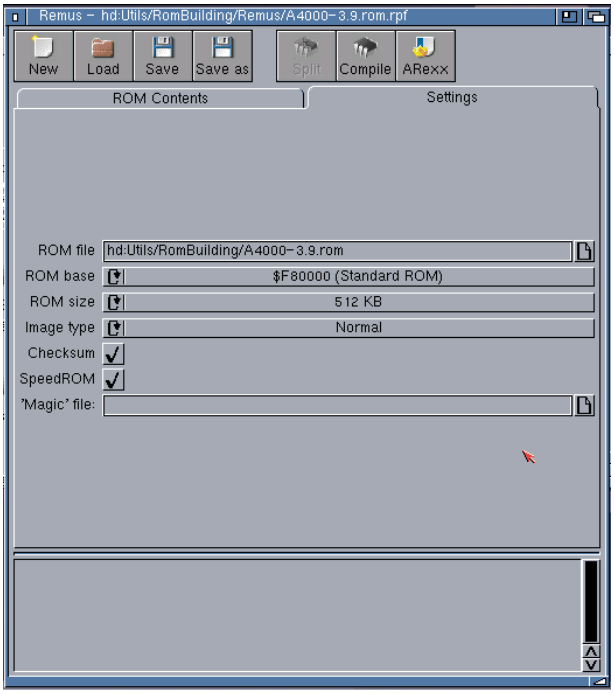
```
$000000-$1FFFFF Amiga  Chip Memory
$200000-$9F0000 Zorro II Memory Expansion Space
$A00000-$B7FFFF Zorro II I/O Expansion Space
$B80000-$BEFFFF Reserved
$BF0000-$BFFFFF CIA Ports Timers
$C00000-$C7FFFF Expansion Memory
$C80000-$D7FFFF Reserved
$D80000-$DBFFFF Reserved
$DC0000-$DDFFFF Memory Mapped Clock
$DD0000-$DEFFFF SCSI Control
$DE0000-$DEFFFF Motherboard Resources
$DF0000-$DFFFFF Amiga  Chip Registers
$E00000-$E7FFFF Reserved
$E80000-$EFFFFF Zorro II I/O  Configuration
$F00000-$F7FFFF Diagnostic ROM (Reserved)
$F80000-$FFFFFF High ROM (512K)
```

Burada, iki bölge dikkatinizi çekebilir, $F8 0000 - $FF FFFF ve $E0 0000 - $E7 FFFF aralıkları. Bunlardan $F8 0000 - $FF FFFF kickstart ROM'un yerleştiği bölge olurken, $E0 0000 - $E7 FFFF, kickstart genişlemeleri için boş bırakılmış bir bölgedir. CD32 ve CDTV sistemleride ROM'larındaki ekstra bilgileri bu adreslere yerleştirirler.

Kickstartımız için ekstra bir 512K adres bulduk, peki burayı nasıl kullanacağız ? Buraya yerleştirdiğimiz rom modülleri otomatik olarak sistem tarafından görülecek mi ? Bu sorulardan ikincisinin cevabı, hayır. Extended rom space dediğimiz bu alanı kullanmayan Amigalarda rom alanı exec.library'de $F8 0000 - $FF FFFF aralığı olarak sınırlanmıştır, sistem bunun dışında bir aralıktaki rom modüllerini aramaz. CDTV ve CD32 gibi sistemlerde ise $E0 0000 - $E7 FFFF aralığını da aktif hale getiren özel bir exec.library bulunur. Dolayısıyla yapmamız gereken ya bir CD32 (32 bit sistemler için), CDTV (16 bit sistemler için) exec.library'si kullanmak, yada yine doobrey'in hazırladığı herhangi bir exec.library'i modifiye eden 1mb patch'ini kullanmak.

1 mb patch'i ne yazık ki standart Remus programında çalışmıyor. O yüzden http://www.doobreynet.co.uk/beta/Remus_1-rc7.lha adresinden beta sürümünü indirip kullanmak durumundasınız. $F8 0000 - $FF FFFF ve $E0 0000 - $E7 FFFF adresleri parçalı olduğundan, romumuzu iki ayrı bölümde hazırlayacağız. İlk bölüm, exec library'mizin olduğu ana bölüm olacak, ve remus'da $F8 0000 adresinden başlayıp 512K yer kaplayan bir alan ayıracağız. İkinci bölüm ise, ek bölümümüz olacak ve burası için $E0

0000 adresinden başlayıp 512K yer kaplayan bir alan ayıracağız. Exec.library'miz ilk bölümde olacak, ve buraya 1megrom patch'inide yerleştireceğiz. İkinci bölüm ise romheader modülü ile başlayacak. Ondan sonra dilediğimiz modülleri buraya yerleştirebiliriz.



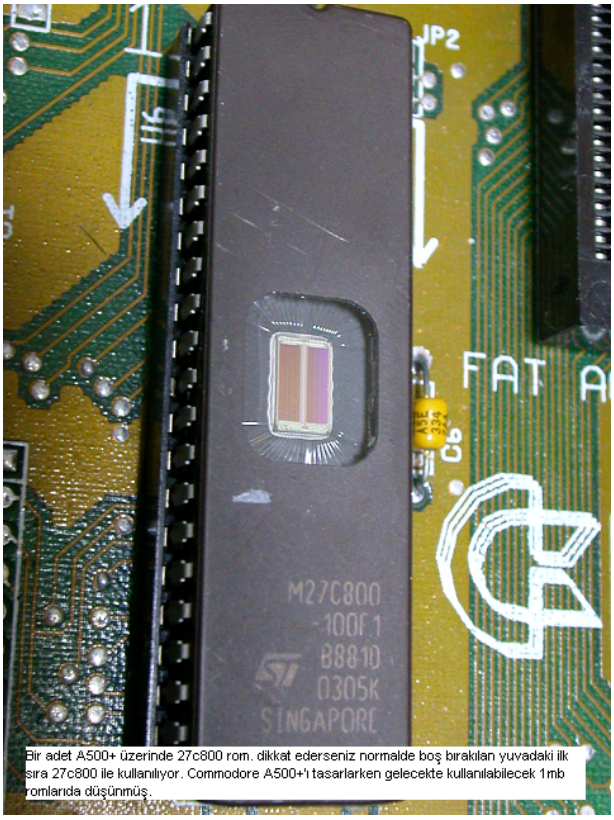
Remus'ta seçenekler bölümü. Extended rom hazırlarken burada Rom Base'i $E00000 yapmanız gerekiyor.

Şekil 11. Remus'ta seçenekler

Bu şekilde normal romunuzun normal ve ek bölümlerini hazırladıktan sonra winuae ile deneyin. Winuae'de rom kısmında, main rom file kısmına normal bölümü, extended rom file kısmına ise ek bölümü koyun. Bu şekilde boot ederek ek bölümdeki modüllerin yüklenip yüklenmediğini kontrol edin.

Eğer romunuz winuae testinden geçtiyse yazılmaya hazır demektir. Yazım aşamasında 32 bit sistem romlarını (Amiga 1200, 3000, 4000) yine 16'şar bitlik iki parçaya ayırmamız gerekecek. 16 bit sistemler için ise yine bu aşamaya gerek yok. Winhex'i açın ve elinizdeki iki rom dosyasınıda dissect 16 bit ile ikişer parçaya ayırın. Bu işlemi yaparken dosyalara ex1, ex2, nor1, nor2 gibi isimler verin, elinizde dört adet dosya olmalı. Bundan sonra ise yine biraz kafa karıştırıcı bir manevra yapmamız gerekiyor. Elimizdeki iki rom'u birleştirip tek bir rom haline getireceğiz. Ama bunu yaparken önce ek bolümü (extended rom file) sonra ise normal bölümü (main rom file) almamız gerekiyor. Bunun sebebi ise amiga'nın boot ederken extended rom'u daha önce görmesi. 16 bit romlar için yapmanız gereken, concatenate diyip hedef dosyayı belirledikten sonra önce extended rom'u, sonra main rom'u seçmek ve işlemi bitirmek. 32 bit sistemler için ise daha önce ikişer parçaya ayırdığınız rom dosyalarını yine önce extended ve sonra normal parçaları olacak şekilde birleştirmek ve iki rom dosyası elde etmek. Yani concatenate diyip son1 diye bir dosya yaratın. Bu dosyaya ex1 ve nor1 dosyalarını concatenate ettikten sonra done ile bitirin. Elde ettiğimiz bu son1 dosyası amiga 1200'de u6a soketine gidecek rom'a yazılacak. Aynı şekilde son2'yi oluşturun, bu dosyada u6b soketindeki rom'a yazılacak.

Bu noktada 16 bit sistem kullanıcılarının karşısına bir sorun çıkacak. 32 bitlik sistemlerde 27c400 romları iki adet 256k'lık parçanın üst üste yazılmasına uygun, zaten yazının başında hatırlarsanız 512K lık bir rom kopyalarken aynı dosyayı iki kere üst üste kopyalamıştık. 16 bit sistemlerde ise 27c400 rom iki 512K'lık parçaya yetmeyecektir. Onun için yazarken 27c800 gibi bir rom kullanmamız gerekiyor. Ayrıca amiganızda bu romu kullanabilecek ekstra adres pinin olup olmadığını kontrol edin. Amiga 500+ kullanıcıları rom soketleri 42 pin olduğundan rahatlıkla kullanabilir. Amiga 2000'lerde ise ECS sistemlerde 42 pinlik bir soket olabilir, ama kesin emin olmak için kontrol etmenizde fayda var.



Bir adet A500+ üzerinde 27c800 rom. dikkat ederseniz normalde boş bırakılan yuvadaki ilk sıra 27c800 ile kullanılıyor. Commodore A500+'ı tasarlarken gelecekte kullanılabilecek 1mb romlarıda düşünmüş.

Şekil 12. A500+ üzerinde 27C800

Bu işlemleri yapıp dosyalarınızı oluşturduktan sonra yazabilirsiniz. yazarken yine swap byte işlemini yapmayı unutmayın.

Şu an için amigalarımızdaki bu ek rom bölgesine koyabileceğimiz modüller sınırlı. Ama yeni modüller üzerinde çalışmalar devam ediyor. En heyecan verici olan gelişme ise bir action replay klonu olan HRTMon monitör programının Winuae programcısı Toni Willen tarafından rom modülüne çevrilmiş olması. Bu modül sayesinde amiganızda ek bir kartuşa ihtiyaç duymadan action replay tarzı bir monitör kullanabileceksiniz. Kulla-

nabilmek için ise amiganıza basit bir hack ile bir level 7 interrupt switch'i yapmanız gerekiyor. HRT modülü sadece bir level 7 interrupt durumunda devreye giriyor. Bu konuyla ilgili daha ayrıntılı bilgiyi EAB forumunda bulabilirsiniz.

Sonuç

Kickstart rom hazırlama ile ilgili söylenebilecekler şimdilik bu kadar. Yazıyı bitirmeden önce size birde ufak ödev vermek istiyorum. 27c800 kullanarak amiga 500/600/2000 için bir kickstart switch yapabiliriz. Nasıl olabilir bir düşünün :). Takıldığınız yerler olursa forumumuzda [http://www.commodore.gen.tr] bize danışabilirsiniz, hepinize iyi programlamalar !

Türk Scene Tarihi

Uğur 'Vigo' Özyılmazel

Giriş

Merhaba sevgili Plazma okuyucuları! Bu aydan itibaren sizlere Scene dünyasının Türkiye ve Türkler ile ilgili kısmını yazacağım. Internet üzerinde yaptığım aramalara rağmen konu ile ilgili detaylı bilgi bulamadım. Amacım, gelecek nesillere en azından belge tadında bir doküman bırakmak!

Öncelikle Scene nedir? Kısaca tanımını yapmak istiyorum. Bizi sözlük anlamı çok da fazla ilgilendirmiyor. Önemli olan, kullanıldığı yerdeki anlamı. Normal şartlar altında her konunun kendine ait bir Scene'i olabilir. Yani kedileri seven, zaman zaman biraraya gelen, yeni çıkan kedi mamaları hakkından konuşan, konu ile ilgili web sitesi açan ya da “en güzel kedi” yarışması düzenleyen insanların oluşturduğu camiaya da Kedi Scene'i ya da Kedi Sevenler Scene'i denebilir.

Bu scene içinde olanlar, "en değerli kedi türü hangisidir", "en nadir bulunan kedi rengi", "en şişko kedi" ya da "kediler en çok neyi yemeyi sever" gibi acayip konuları çok iyi bilirler. Hatta var ise kediler ile ilgili belgesel dvd'leri, aylık kedi magazinleri gibi şeyleri de satın alırlar. Yani kedi konusu ile ilgili her olaya girer, her tür medyayı takip eder, kendisi gibi ilgili diğer insanları bir şekilde bulur, tanışır ve hayatı boyunca, paso bu konularda sohbet etmek ister.

İşte bizim Scene de aynı bunun gibi birşey. Kronolojik olarak baktığımız zaman, Scene 80'lerin başları ve ortası gibi başlıyor. O yılların en popüler bilgisayarları olan Commodore 64, Atari, Spectrum ve Amstrad, Scene'in doğmasına yardımcı oluyor. Tabii ki en yaygın ve dolayısıyla en domine sistem Commodore 64.

İlk etapta oyun kırma işleri Scene'inin temel unsuru oluyor. Oyunların olduğu teyp kasetlerinde, kopyalamaya karşı koruma bulunuyor (Copy Protection). Bu koruma zahmetli bir kurcalama sonucu ortadan kaldırılıyor ve oyunlar artık kopyalanabilir hale geliyor. Daha sonra, teypten ara-yüklemeli oyunlar kimi zaman tek parça dosya haline, kimi zaman da disketten yüklenebilecek hale getiriliyor. Bu işleri yapan insanlar bir anda kendilerini “Crack Scene” içinde buluyorlar... Bu işleri yapana “Cracker”, orijinal oyunu bulana (genelde satın alana) “Original Supplier” deniliyor.

Bu işleri yapmak kolay değil. Hem zor hem de uzun vakit alan bir işlem. İnsanın doğasında olan, ünlü olmak - tanınmak iç güdüsü, bu işi yapan insanların kendini tanıtma isteğini doğuruyor. Oyun kıranlar, kırdıkları oyunların başına kendi takma adlarını ya da bu işi yaptıkları ekibin adını (Grup) yazmaya başlıyor. Böylece oyunu kıran kişi/grup artık tanınmaya başlıyor (tabii ki takma isimlerle).

Böylece “Intro” dediğimiz kavram doğuyor. Oyunların başına koyulan, "introduction" kelimesinin kısaltması olan “intro” lar, oyunu kıran grubu tanıtıyor. Yapılan crack ile ilgili notlar, eğer oyun "train" edilmişse (sonsuz hak, sonsuz mermi gibi) trainer ile ilgili notlar vs. Asıl önemli bilgi ise bu gruplarla temasa geçmek için bırakılan adres ve telefonlar.

Yavaş yavaş iş büyüyor, özellikle Avrupa'nın pek çok ülkesinden gruplar türemeye başlıyor. İnternetin olmadığı bu dönemlerde insanlar birbiriyle posta yoluyla haberleşiyor. Disketler gidip geliyor tüm dünyada. “Swap” dediğimiz olay doğuyor. Teknoloji ilerliyor, “Modem” denilen efsanevi cihaz çıkıyor. Artık crackerlar mektup / posta yerine direk uzaktan bilgisayarlarını birbirine bağlıyor. İlk başlarda inanılmaz düşük hızlarda da olsa bu, "Modem Scene'inin" doğmasını sağlıyor.

Zaman içinde gruplar, intro'larına daha çok önem vermeye başlıyor. İlk başlarda çok basit tasarımlara sahip olan introlar, yavaş yavaş özenle yapılmış grup logolu, çok renkli karakterlerle yazılmış yazılı mesajlara sahip oluyor. Intro'larda çalan müzikler ciddi ilerleme kaydediyor, ekranda sabit duran yazılar birden kaymaya başlıyor.

Gruplar zaman zaman biraraya geliyor, bilgisayarlarını yanında getiriyor ve oyun kırma partileri yapmaya başlıyor. “Copy Party” denilen şey doğuyor. Yarışmalar yapılıyor "ilk şu oyunu kim kıracak" , "en fazla trainer opsiyonunu kim ekleyecek?" gibi. Bunlar da yetmiyor, artık "en güzel intro'yu kim yapacak", "en güzel logolu intro kimin olacak" gibi saplantılar başlıyor.

Görülüyor ki, oyunu kırmaktan daha zevkli şeyler var. Introlar oyun başlarında çıkan küçük şeyler olmaktan kurtulup, tek başlarına yayınlanan minik ürünler haline geliyor. Bu da yetmiyor, 4-5 tane intro arka arkaya paketlenip mini bir intro-koleksiyonu oluyor. Artık bunların adı “intro” dan çıkıp yavaş yavaş “demo” adını alıyor. Demonstration kelimesinin kısaltması yani.

Bu noktada Scene, kendi içinde küçük bir ayrıma gidiyor, Scene ve Demoscene şeklinde... Tüm illegal işler (oyun kırmak , kopyalamak , kaçak telefon hatları gibi) Scene'e ait oluyor, Demo / Intro gibi sanatsal ağırlıklı işler ise Demoscene'e ait oluyor. Bunun dışında, “BBS Scene” , “Crack Scene”, “Hack Scene”, “Asci Scene” gibi değişik fraksiyonlar da doğuyor.

Bu bölünmeler platform / makine bazında da oluyor. “Amiga Scene”, “C64 Scene”, “Atari Scene”, “Amstrad Scene” gibi o dönemin popüler makineleri, kendine özgü bir scene'in parçası haline geliyor.

Benim kendi içinde bulunduğum ve 1988 yılından beri bizzat takipcisi olduğum platformlar C64 , Amiga ve bazı Console/ Handheld cihazlar ile ilgili scene'ler. Elimden geldiği kadar PC scene'ini de takip ediyorum. Fakat benim için gerçek scene hep C64 ve Amiga scene'i olmuştur.

Bu uzunca Scene tanımlamalarını yaptıktan sonra, asıl konumuz olan “Türk Scene”i ne geçelim artık.

İlk Türk Grupları

İlkleri yazmak ve anlatmak gerçekten çok güç. Sebebi şu, benim scene ile tanışmamdan öncesini iyi bilmem gerekiyor. Ne yazık ki bugün (2008) itibari ile, 1980'li yıllardan kalan scener sayısı yok denecek kadar az. Pek çoğunun sadece takma adını bili-

yoruz, belki bir kısmını da hiç bilmiyoruz. Umarım bu yazı dizisini o eski güzel günlerden kalan abilerimiz okur da bize bir şekilde ulaşır, hatalarımızı eksiklerimizi düzeltmemize yardımcı olur.

1988 yılında, mahalle ve grup arkadaşım olan sevgili Emre Yavuzkal (ilk adı demo idi sonra skywalker oldu) ile Amiga 500'de oyunlara bakarken hep karşımıza “Zombie Boys” adlı bir Türk grubunun introları çıkardı. Keza, ben, C64'ümdeki oyunlara bakarken karşıma hep “The Metro Boys”, “Fast Generation” , “The Joker Crew”, “Comrade”, “The Danger Cobra” gibi grupların introları çıkardı.

İlk başlarda bu grupları, adlarından ve introlarda geçen ingilizce yazılardan dolayı yabancı zannederdim. Meğerse öz be öz yerli malı yurdun malı tarzı bizim öz Türk gruplarıymış hepsi.

Şekil 1. Bir Joker Crew introsu

Küçük bir dip not, aslında keşke İstanbul dışından, diğer şehirlerde oturan arkadaşlarımız da bize ulaşsa da, o dönemlerde o şehirlerde acaba hangi gruplar vardı bilsek. Benim yazacağım bu belge, ne yazık ki yaşadığım yer ile bağlantılı. İstanbul'u baz almak zorundayım. İleriki yıllarda İzmir, Ankara gibi diğer şehirlerdeki gruplarla ve arkadaşlarla temasa geçtik, fakat işin ilk başlarında o şehirlerde durumun ne olduğunu ben de tam bilemiyorum.

Konuya geri dönelim, o yıllarda karşımıza çıkan C64 gruplarını listelersek;

1. The Metro Boys
2. Fast Generation
3. Turkish Mega Force
4. Light Force
5. The Joker Crew
6. Comrade
7. The Danger Cobra
8. TNF'77
9. The Cobra Boys
10. Echo Crew (Ankara)
11. Pet Shop Boys (Ankara)

Bu gruplar genelde “Import” grubu dediğimiz, yurt dışı ile kontağı bulunan, özellikle oyun ağırlıklı swap yapan, gelen oyunların başındaki orijinal intro'ları çıkarıp kendi introlarını koyan ve bilgisayar dükkanlarına dağıtan gruplardı.

Şekil 2. Bir Metro Boys introsu

Bu gruplar içinde , kendilerine özgü introları bulunan (genelde müzik ve fontlar başka yabancı gruplara aitti çünkü o zamanlarda grafiker ve müzisyen bulmak neredeyse imkansızdı) gruplar The Metro Boys, The Joker Crew, Comrade, The Danger Cobra idi. Diğer gruplar “Ripping is an Art” (yani başkasının yaptığını araklamak bir sanattır) deyişini hayat felsefesi yapmışlardı ve pek çok yabancı grubun introsunu araklayıp kendi logolarını koyup sanki kendi introları gibi dağıtmışlardı. (Bunu yapanlardan biri de bendim, detaylar ileriki paragraflarda)

O yıllarda biraz yabancısı olduğum Amiga Scene'inde ise yine gruplar türemiş ve aynen C64'te olduğu gibi release'lere devam etmişlerdi. İşte hatırladığım Amiga grupları;

1. Zombie Boys
2. Tacs From Turkey
3. Angels
4. ACS (Ahmet Cracking Service)
5. Bodys

1988 yılında kendi grubumuz olan “The Hacker Boys” u kurmuştuk. C64 ve Amiga platform'unda işler yapıyorduk. Özellikle arakladığımız C64 introlarını sanki kendi release'lerimizmiş gibi bilgisayarcılara dağıttığımız oyunların başına koyuyorduk.

Commodore Show 1988

O yıl düzenlenen Commodore Show'a gitmiştim. Elimde 1-2 tane intro-maker (hatta intro-packer) stand stand dolaşıp “Intro Maker var mı?” diye soruyordum. A-Soft adlı dükkanın standına girdiğimde, ilk kez birisi bana “Evet intro maker var” demişti. Ben de “Bende de var, değişelim mi?” dedim. Cevap “Yok, istersen satarım” şeklinde olmuştu. O güne kadar topu topu 4-5 tane disketim vardı. Evet, taa ki o güne kadar.

Bana intro satmak isteyen kişi ile bir anda kanka olmuştuk. Az ileride bulunan Teleteknik standında özel C64 köşesi vardı. 7-8 tane C64 set insanlar kullansın, takılsın diye duruyordu. 2-3 tane setin önü inanılmaz kalabalık. Birisi bağırıyor “Coca-Cola demosunu isteyen var mıı?” millet hemen disketini uzatıyor anında kopya... Yani Commodore Show biranda Copy-Party tadına bürünmüş.

Şekil 3. Bir Zombie Boys introsu

Disketim olmadığı için koşarak o intro satmak isteyen arkadaşa gittim, durumu anlattım, bana 1 kutu parrot marka disket (1 kutu = 10 disket) verdi. 5 dakika sonra yanına geri döndüm , disketler ful dolmuştu. Yeni bir kutu istedim. O gün fuarın son günüydü, akşam saat 18:00 olmuştu, “kapatıyoruz” anonsu yapılırken biz halen disket kopyalıyorduk. O günden sonra yaklaşık 200 tane disketim olmuştu. Kaç kutu kopyaladık hatırlamıyorum bile... Parti yarım kalmış, ertesi gün, oradan tanıştığımız bir arkadaşın evinde kopy'e devam etmiştik.

O gün evine gittiğim o arkadaş ile yıllar sonra Show TV binasında karşılaştım. Kendisi Digiturk'ün Genel Müdürü olmuştu :) Neyse, o gün, kopya için gelen arkadaşların biri, C64'de ufak tefek Assembler yazıyormuş, bana ekrana yazı yazdırmayı göstermişti... O an da benim için dönüm noktası olmuştur.

O kadar çok bakacak intro / demo vardıki anlatamam...

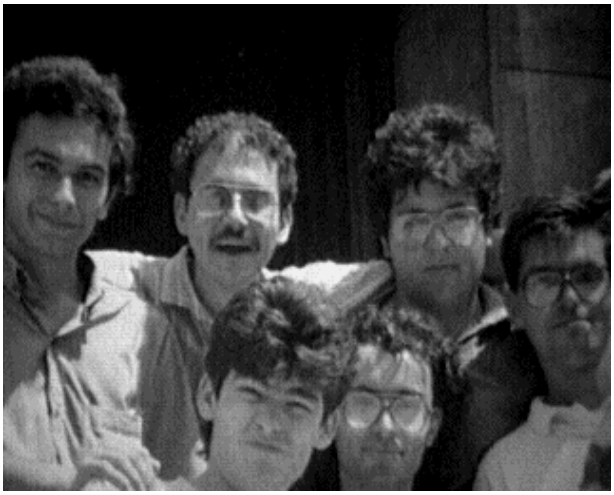
Violet Osman ve Angels'dan Süha

Bir gün, Suadiye/İstanbul'da bulunan (halen de açık olan) Violet Computer House adlı yerde, grup arkadaşım demo, sinclair oyunu satarken, içeri biri girdi. Dükkan sahibi Osman abi, “Bak işte oyunları bunlar getiriyor” dedi bana. Ben bir anda içeri giren kişiye heyecanla baktım, hemen atladım tabi, “Abi kimsiniz nesiniz? nasıl oluyorda oluyor” gibi. Meğer karşımdaki kişi Angels grubundan Süha imiş.

Sağolsun, bana bir disket verdi, dedi ki “Bak bunun içinde diskmag denen programlar var, orada değiş-tokuş (swap) yapmak isteyen kişilerin ve grupların adresleri var, İngilizcen varsa onlara mektup yaz, bir de disket doldurup gönder” dedi. Belkide Scene'e girişime ön ayak olan bu kişi, yıllar sonra benim için düşman statüsünde biri olacak, rakip grup olacaktık. Tabi kişisel bir şey değil, grup olayları her zaman herşeyin üzerinde :) :) :)

Eve gidip diske baktığım zaman aklım durmuştu. Party Invitation'lar (Copy Partiler için davetiye introları), yeni demolar, introlar, disk-mag'ler... O gün benim hayatımda bir mihenk taşı oldu.

Fuckings to Zombie Boys



arka(soldan sağa): move/zombie boys , creep/tacs from turkey, turbo/zombie boys, ???, ön : jason/zombie boys, malice/zombie boys

Şekil 4. Zombie Boys ve TACS

The Hacker Boys olarak, her oyunun başında çıkan Zombie Boys introları bizi çok fena uyuz etmişti. Biz de o günden itibaren işi gücü bırakıp dağıttığımız oyunlara koyduğumuz tüm introlarda Zombie Boys'a küfür ettik. Reklamın iyisi kötüsü olmaz değil mi?

Grubumuz büyümüş ve şu üyeleri bünyesine katmıştı : Ventor, Evilman, Spiderman, Nighthawk, Eldorado, Demo. Şaka maka 6 kişiydik. Ventor ve Evilman C64'de code yazıyordu, Spiderman C64'de grafiker di. Nighthawk Amiga'da grafik müzik, Demo Amiga'da code ve Eldorado'da Amiga'da swapper dı ve Erman Amiga House'da çalışıyordu :)

Ed: Şimdiki takma adı ile Spaztica. Yukarıdaki resimdeki Jason'ın bugunkü hali yani. Yıllar karizmayı artırmış :)

Şekil 5.

Birgün Nighthawk gelip bize “Ben Zombie Boys'a katıldım” dedi. Bizi arkamızdan bıçakladı yani... O dönemlerde Zombie Boys, tüm grup olarak Aytaç Bilgisayar / Mecidiyeköy / İstanbul'da takılırdı. Grubun efsane coder'ı Move, Aytaç Bilgisayarda çalışırdı. Bir gün Nighthawk ile beraber Zombie Boys toplantısına ben de kaçak olarak gitmiş, gittiğim gibi de benim The Hacker Boys olduğumu anında anlamışlardı. “Neden bize küfür ediyorsunuz” diye soran Turbo'ya “Ben etmiyorum, bizim Amiga'cı ediyor” demiş ve bir şekilde olaydan sıyrılmaya çalışmıştım :) :) :) (Ed: Yalnızca bu hikayenin bütün detaylarını canlı olarak Vigo'dan dinlemek için bir demo partisine gitmeye değer :))

The Joker Crew

Zombie Boys toplantısından kazasız bir şekilde kurtulmuştum. En azından atık onlar da The Hacker Boys'u biliyordu. Benim de yurt dışı ile bağlantım olmuş, oyun olsun, demo olsun, dergi olsun yağmur gibi gelmeye başlamıştı. Commodore dergisinde “Compushop” adlı bir dükkanın ilanını gördük. İlanda, “Intro Maker satılır” yazıyordu. Büyük bir heyecanla Bostancı'dan kalkıp hayatımda ilk kez Merter adındaki semte gitmiştim diğer grup arkadaşımla.

Tonlarca intro maker. Meğerse Compushop'un sahibi aslında Türk C64 gruplarının en önemlilerinden olan “The Joker Crew” un founder'ı (kurucusu , lideri) imiş. Bu sayede pek çok diğer Türk grupları ile yüz yüze tanışma şansım oldu. TNF'77, The Cobra Boys gibi...

Zaman içinde The Hacker Boys'u kapatıp biz de The Joker Crew'a katıldık. Düzenli olarak swap yapıyorduk. En son çıkan demolar, oyunlar, disk-mag'ler, hepsi geliyordu yurt dışından. TJC sayesinde pek çok yabancı grupla iletişime geçtik. Oyun dağıtımı konusunda fırtına gibi esiyorduk.

Zombie Boys ve Bronx'un kurucusu ve dünyanın en iyi grafikerlerinden biri: Turbo

Şekil 6.

Bu esnada diğer Türk grupları genelde kendilerine üs olarak bir bilgisayarcı seçerler ve orada takılırlardı. Genelde burada çalışan 1-2 kişi de o grubun üyesi olurdu. Grup o dükkana bedava olarak oyun verir, bunun karşılığında hem orada takılır hemde disket ihtiyacını karşılar, arada beleşe yemek yer, gazoz içerdi.

Erman Amiga House

İşte bu tarz dükkanlardan en meşhuru, başta Feneryolu/İstanbul'da olan, daha sonra Kadıköy/İstanbul'a taşınan Erman Amiga House (yada Erman Software House/Erman Elektronik) idi. Bu dükkan, dönemin en ünlü gruplarına ev sahipliği yaptığı gibi, Commodore Dergisi'nin en ünlü yazarlarının da takıldığı hatta çalıştığı mekandır. Ayhan Kalaylıoğlu, Tuna Ertemalp gibi efsanevi Developer'lar hep bu dükkanın tayfasındandır (Ed: Tuna Ertemalp şu an ABD'de yaşamaktadır ve Microsoft'ta üst düzey bir yönetici olarak çalışmaktadır).

Keza, Crockett/Metro Boys ^ Zombie Boys, Caiser Boys/Zombie Boys da bu dükkanda çalışan elemanlardandır. Yani Erman'a gelirseniz kesin ünlü bir scener'ı görme şansınız vardı. Ben de bu ümitlerle Erman'a takılırken Comrades / The Danger Cobra grublarının coder'ı Mephisto ile tanıştım. Kendisi o an için Zombie Boys'un C64 bölümünün lideri imiş. Başka bir dükkanda buluşmak için randevüleşmemize rağmen ben o gün oraya gidememiştim. O devirde cep telefonu da yoktu :) Bir şekilde kontağı kaybetmiştik.

1989 yılına geldiğimizde, ben üniversiteye başlamıştım. Yan sınıfımız Fizik bölümüydü ve fizik bölümünde 2.sınıfta okuyan bir scener ile karşılaşmıştım. Bu scener, bir yıl önce buluşmayı başaramadığım Mephisto/Zombie Boys'du.

Grup Sayısı Artıyor

1989 yılının sonlarına yaklaştığımızda, ağırlıkta Commodore 64 olmak üzere irili ufaklı gruplar türemeye başladığını görüyoruz. Bu gruplar genel olarak o dönemde import grubu. Yani grup olarak kendilerine ait prodüksiyon üretmeyen gruplar (bazıları daha sonra kendi ürünlerini de üretecekti). İlk bakışta aklıma gelen gruplar;

1. Accuracy
2. Chaos #1
3. Medal
4. Uğur Moda
5. Bross

İşin garip yanı, bilgisayar dükkanları da bir grup gibi takılıyordu o dönemlerde. A-Soft, Sorbim , Abacus, Erman , Laçin, UFO gibi dükkanlar, derme çatma introlar ile (genelde maker yardımıyla yapılmış) kendi çaplarında release yapıyorlardı.

Önümüzdeki ay Türk gruplarının yükselişini, Scene'den Demoscene'e geçişlerini anlatacağım. Yazıda gördüğünüz eksik / yanlış / hatalı ne var ise lütfen bana iletin.

Plazma Künye

Plazma

Amatör Bilgisayar Kültürü

http://plazma.tr-demoscene.info

Sayi 4 : 5 Şubat 2008

Ekip

Yazarlar:

- Ahmet Moldibi (Datura/Glance)
- Arda Karaduman (CoZe)
- Arda Ö. Erdikmen (Ref)
- Atilla Filiz
- Bilgem Çakır (Nightlord/Glance^Aesrude)
- Ege Şafak Erkul (Punky/Naked Hamsterz)
- Emir Akaydın (Skate/Glance^Clash)
- Emir Diril (Drey/Demodojo)
- Emirhan Bayyurt (Ragnor/Clash)
- Hüseyin Kılıç (Wisdom/Crescent)
- Kürşad Karamahmutoğlu (Hydrogen/Glance)
- Özgür Yıldırım (Lord Henry Wotton II)
- Şemseddin Moldibi(Endo/Glance)
- Türker Gürevin(Alcofribas)
- Uğur Özyılmazel (Vigo/Bronx)

Kapak:

- 2D/3D art: N. Burak Yetgin (Hyper/Bronx)
- 2D art ve layout: Kürşad Karamahmutoğlu (Hydrogen/Glance)

Yardımcı Editörler:

- Emirhan Bayyurt (Ragnor/Clash)
- Kürşad Karamahmutoğlu (Hydrogen/Glance)

Editör:

Bilgem Çakır (Nightlord/Glance^Aesrude)

İletişim:

Plazma'da yazar olmak istiyorsanız veya dergi ile ilgili görüşlerinizi bizimle paylaşmak isterseniz, aşağıdaki adres aracılığıyla editörlerle temasa geçebilirsiniz.

nightlord (at) nightnetwork (nokta) org