Plazma 8

Plazma 8

Yayımlanma

İçindekiler

Editörden 1
Plazma'nın Bu Sayısında 2
Caisson'un 7dx-2010 Raporu 3
Drey'in 7dx2010 Raporu 5
Nightlord'un 7dx-2010 Raporu 8
LW3D'nin 7dx-2010 Raporu 14
C++ Kursu 4 16
SDL Kursu 1 20
Grafik Efektler için Matematik 3 27
C++'da Akıllı İşaretçiler 32
Alternatif Gerçeklik Yaratmak 36
Alternatif Dünyalarda Toplumlar ve Tarihleri 39
Öz Desert Dream - Perde Arkası 42
7dx 2010 Ürün İncelemeleri 49
Plazma Künye 63

Editörden

Bilgem 'Nightlord' Çakır

Aynı anda üç sayıyla muhteşem bir dönüş yapan Plazma dergisinin sekizinci sayısına hoş geldiniz. Bu satırları yazdığım sırada son haftalardaki uzun ve yorucu bir sürecin ardından, Plazma'nın 6. 7. ve 8. sayıları karşımda duruyor. Pdf versiyonları toplamda 200 sayfayı geçen, içi programlama, grafik, gibi dallarda eğitici yazılardan, eğlenceli parti raporlarına kadar her tür içerikle dolup taşan uzun süre okuyup bitiremeyeceğiniz bu üç sayı sonunda okuyucuya ulaşmak üzere.

Bunun sonunda geç de olsa gerçekleşiyor olması bende büyük bir mutluluk yaratıyor. Evet çok geciktik, ama yok olmadık, buradayız. Geçtiğimiz üç yılda bir süre ulaşılamaz hale düşen web sitemizden tutun da, okuyuculardan gelen haklı sitemlere kadar, Plazma zor günler geçirdi. Tıpkı Türk scene'inin genelinin dönem dönem yaşadığı problemler gibi, Plazma da pekçok insanın kafasında "bitti mi" sorusunu tetikledi.

Ama bitmedi. Dayandı, bu fırtınaları da atlattı ve burada karşınızda. Türkiye'deki amatör bilgisayar meraklılarının yanında geçmişte olduğu gibi bugun de varız. Siz yeter ki birşeyler üretmek isteyin. İhtiyaç duyacağınız bilgi ve desteği size vermeye devam edeceğiz.

Çünkü öğrendik ki bu alemde en büyük güç "devamlılık". Plazma gereken devamlılığı göstererek, içerik oluşturmaya devam etti ve bugün burda. Amatör bilgisayar dünyasının bütün üyeleri de hobilerinde ve uğraşlarında devamlılık gösterdikçe başarılı olacaklar, var olacaklar.

Bu arada küçük bir parantez açıp 2011 baharında olan mutlu bir olayı da burada paylaşmak istiyorum. Bu editörden köşesi için resim ararken harddiskimde bir resme rastladım. Glance grubunun Snaphshot demosunu yapan ekibinin Breakpoint 2009 partisinde çekilmiş resmi. Bu resim, bana derginin 8. Sayısında 2011'de olan ve bizi süper mutlu eden bir olayı Plazmada kısa da olsa paylaşmak gerektiğini hatırlattı. Snapshot demomuz hatırlayacağınız gibi 2010 yılında dünyanın en büyük demo partisi Breakpoint'te C64 demo yarışmasını kazanmıştı. Bu demo 2011 Baharında yapılan Scene.org Ödüllerinde de Yılın "Retro Platformlardaki En İyi Demo" ödülünü aldı. Ayrıca "Halkın seçimi" kategorisinde (normalde diğer kategoriler bir jüri tarafından belirleniyor) finale kalan tek 8 bit demo oldu.

Bunda da Türkiye'deki scenerlar ve amatör bilgisayar kültürü tutkunlarının büyük desteği rol oynadı. Bunun için oy kullananlara da ne kadar teşekkür etsek azdır.

Bunun sonucu olarak işte o resim bu sayfada yerini de aldı :)

Şekil 1.

Ki dergiyle çok da ilgisiz sayılmaz. Resimdekilerden üçü derginin editörleri ikisi yazar vs :)

Şimdi aklınızda Plazma'nın yeni sayısını ne zaman göreceğinize dair bir soru olabilir. Bu sorunun net bir cevabı yok. 4. ve 5. sayılarda 3 ayda bir yeni sayıyı çıkarmak hedefini koymuştuk. 6. 7. Ve 8. Sayılar bize öğretti ki amatör ruhla hazırlanan bir dergide bunu başarmak ya mümkün değil ya da bu süreçle gerçekleştirilemiyor.

Bu yüzden önümüzdeki dönemde yeni bir model uygulayacağız.

Her Plazma sayısı için editörleriniz bir "başlangıç duyurusu" yapacaklar. Bu duyuru yapıldıktan sonra kısa bir zaman (10 gün) içinde yazarlardan yazılar toplanacak. Ve bu toplanan yazılar hızla birleştirilip (4-5 gün içinde) yeni sayı yayınlanacak. Ebat olarak derginin her sayısının 40 – 60 sayfa aralığında kalmasını bekliyoruz.

Bu verilen süre içinde elimize ulaşmayan yazılar bir sonraki sayıyı bekleyecek. Ancak bir sonraki sayının ne zaman olacağı bilinmeyecek.

Yani artık Plazma periyodik olarak geleceği bilinen (ya da periyodik olmaya çalışırken geç kalan) bir yayın olmayacak. Belirsiz aralıklarda olan bir güzellik olacak. Plazmaya dair kesin olan tek zaman, yazıları davet eden "başlangıç duyurusu" ile derginin çıkması arasında 15 gün olacağı.

Bu sürecin "burst mode" içinde daha verimli çalışan Türk insanı için daha uygun olduğuna inanıyoruz :)

Şimdi 2011 yılının Plazmasıyla sizi baş başa bırakalım.

Plazma'nın Bu Sayısında

7dx 2010 Raporları ve Ürün incelemeleri

ı ve Ürün incelemeleri

2010 yılının son günlerinde yapılan 7dx 2010 partisi ile 7dx serisinin 9. Ayağı tamamlanmış oldu. Yine pekçok dalda amatör bilgisayar ürünlerinin yayınlandığı bu köklü organizasyonun atmosferini 4 parti raporu ile sizlere taşıyoruz. Bunun yanında yazarlarımız bu sayıda partide yayınlanan bütün ürünler hakkında detaylı incelemeler de hazırladı. Zevkle okuyacak, sonraki partilere siz de ürün yayınlamak isteyeceksiniz.

Oyun Yapımı

Oyun Yapımı yazılarımız bu sayıda da devam ediyor. Bu sefer oyun dünyası yaratma konusuna eğilen iki yazı ile sizinleyiz. Kimbilir aranızda bazıları ne dünyalar yaratacak ve o dünyalarda geçen ne oyunlar ne hikayeler ortaya çıkaracak.

Öz Desert Dream - Perde Arkası

7dx2010'un en bomba ürünü olarak wild compoyu kazanan bu süper eğlenceli ürün nasıl hazırlandı? Yapanlar ne zorluklar çektiler? Kuzu kelle nereden bulunur? Kertenkele oyuncağı bulmak ne kadar zor olabilir? Bütün bunların cevabı ve fazlası, partiye damgasını vuran Zomco ekibinden Domino'nun kaleminden geliyor. Okumaya doyamayacaksınız.

Eğitsel Yazılar

Her zamanki gibi bilgi pınarınız Plazma, bu sayıda da teknik yazılarını eksik etmedi. Yeni başlayanlara yönelik C++ kursumuz 4. Bölümüyle gelirken, orta ve ileri seviye C++ kullanıcıları için de Boost ve STL'deki akıllı işaretçileri tanıttığımız bir yazı bu sayıda. Ayrıca Ragnor'un kaleminden çok detaylı bir SDL kursu sizi en baştan alıp ekranda piksel koyacak hale gelene kadar bırakmıyor. Grafik efektler için Matematik yazı dizisi 3. Bölümünde "What is the matrix" sorusunu cevaplıyor.

Kısacası, eğitsel köşelerimiz bir kez daha gösteriyor ki, teknik konuları Türkçe olarak size ulaştırması bizden, bu bilgilerle harikalar yaratmak sizden :)

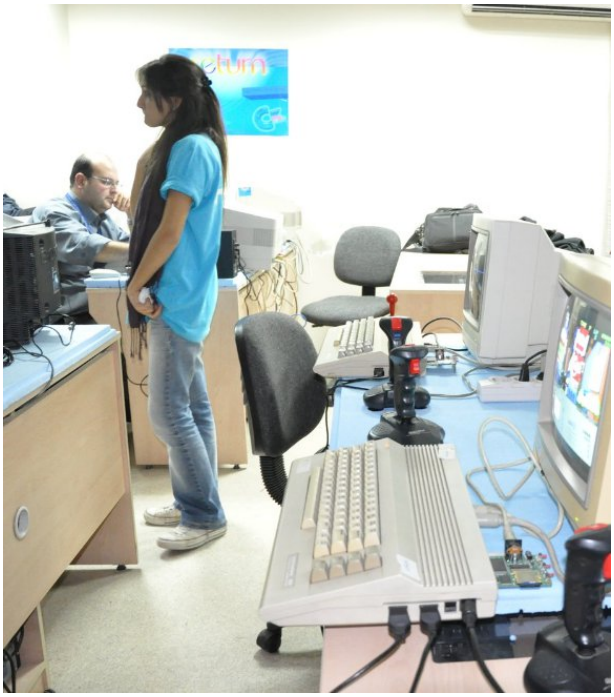
Buyrun karşınızda 8. sayı

Caisson'un 7dx-2010 Raporu

Can 'Caisson' Ulaş

Parti organizasyonunda yer almama rağmen, salak saçma olarak adlandırılabilecek (aslında değil ama öyle) birsürü şeyden ötürü bu sene gerektiği kadar mekanda olup, ilgilenemedim birçok şeyle. Bunun için öncelikle kusuruma bakmayın ve emeği geçen herkese çok teşekkür ederim. Neyse bunu geçelim. Parti ile ilgili 3-5 madde söylemek isterim gözüme ve aklıma takılan. (98'den beri, neredeyse hiç denecek kadar az ortalarda olduğumdan sallamıyor görünüyor olabilirim ama öyle değil tabi)

Cumartesi günkü seminerlere damgayı vuran başlık tabii ki moldibi brothers'ındı. çok keyifli günler yaşamışlar, herkes birbirinden habersizken birşekilde bir yerde temas etmiş falan O sırada sanıyorum ki boun tayfasından bir soru geldi. ilginçti ve güzeldi aslında. "Neden c64'de kaldınız?" "Gönülden bağlı olmak ile alakası çok var" diyerek, çok lezzetli bir cevap verdiler tabii ki. (aslında scene açsından bakıldığında, oradaki duruşun önemliliği, product kalitesi, grup senkronizasyonu, işin içine girildiğinde de "real talent" diye de iki kelime ile de açıklanabilirdi)



Şekil 1.

Bize göre daha yeni jenerasyonun bu konsepti algılamakta güçlük çekmesini ama bir o kadar da merak etmesini hem doğal hem de garip karşılıyorum.

Doğal karşılıyorum, çünkü yeni jenerasyon teknoloji buhranı geçirerek büyüyor (cep telefonları, high-end pc'ler, konsollar, vs vs) ve o makinaları gördüklerinde "aabi bu ne yeaa? çok enteresanmış, müzeden mi falan" tarzında birbirleriyle konuşuyorlar, incelemek istiyorlar ama hafif bir çekince ve "aman abi nesine bakıcam, wiki'den okurum" tadında yaklaşıp one on one, microprose soccer oynayıp geri çekiliyorlar.

Garip karşılıyorum, çünkü merak ettiklerini, ilgilenmek istediklerini dile getirmelerine rağmen, sadece dile getirmekle kalıyorlar, bir adım ötesine geç(e)miyorlar (belki geçiyorlardır, ben denk gelmedim, yanlışım varsa düzeltin)

Cumartesi günü mekandan gece 12 gibi ayrılmak durumunda kaldım, tabii muhabbetleri kaçırdım. Pazar günü de benim için efektif geçti diyemeyeceğim. Demo compoya gelindiğinde maalesef mekandan ayrılmak durumunda kaldım, ama ZOMCO diyorum, helal diyorum

Ayrıca wireless c64 de, benim için ayrı bir dumura yelken açma anıydı. c64'ü hakir görüp fosil muamelesi yapanlara çötönnk diye alüminyum kapağı niteliğinde oturdu.

Bekir'in yeniden bir çalışma ile o kadar sene sonra kendini hatırlatması da benim için ayrı bir noktaydı. Halen eski tadını koruyan şeyler yapıyor olması da hoş

Gelelim bombaya. Return tabii ki. Hepimiz neticede x senedir arkadaşız, joker ve aegis, halen ascraeus'dalar (grup ölmüş bitmiş anasını satim, ehheh) ama bunları bir yana bırakacak olursak, Ben 2011 itibarı ile Return'den güzel gelişmeler bekliyorum. Zaten bu gazı, ben başka kimsede görmedim desem yeridir. Çünkü nightlord'un da dediği gibi kısa denebilecek bir sürede basamaklar return tarafından yardıra yardıra çıkılmaya başlandı. Elimden geldiği sürece de hepsine destek + gazlama konusunda vericem ayarı...

Şimdi diceksiniz ki "lan kelin merhemi olsa başına sürer, otur da kendine bak, kaldır totonu da bıdı bıdı yapacağına güzel birşeylere imza at" Zaten, Hydrogen'in "aabi nooldu palladium? çıkarın artık şu dergiyi, adamı hasta etmeyin" lafı ve "Dr.Razor's Corner diye birşey vardı?!?!?" ironisinden yola çıkarak benim de gazlanma konusunda içerilerde kıvılcımlar olmuyor değil. Umarım bu yeni senede imkanlarımız el verdikçe birkaç şey organize edip gün ışığına çıkarabiliriz Ascraeus olarak.

Zaten bunu düşündüğümde dün gece bir ara fırsat bulup şu anki kafam ile "dur lan, biz 4 sayı çıkardık, neler yazmışız bakalım" diye eski sayıları okudum. Hem çok iyi hem çok kötüydü kendi değerlendirmem olarak.

Şekil 2.

Çok iyiydi, uzun zaman sonra chartlarda ilk 10'a giren bir türk dergisi vardı, ifli kullanan ilk dergiydi (her ne kadar doğru mantıkla çizilmemiş de olsa)

Çok kötüydü, çünkü yer yer gereksiz yazılar vardı, (bazılarını ben bile anlamayıp bu ne lan dedim) içerik sıkıntısı vardı, görmemişler gibi zilyon tane müzik koyup hayvan gibi grafik yükleterek bekleme süresini arttırmıştık.

Zaten son sayıda gözüme çarpan birkaç review vardı bizim dergi ile ilgili.

Resmen okuyup "adamlar haklı beyler" dedim. Ama hiçbir zaman pişmanlık barındırmıyoruz O dönem öyle olması gerekiyordu bilader. Sağolsun son sayıda Digahole'un desteği çok çok yerindeydi. Tekrardan teşekkür ederiz kendisine. O dönemde, bu kadar çömezliğimize rağmen, böyle bir derece yakalamış olduğumuz için ise ayrıca dergide emeği geçen herkese (özellikle Aarcane, abicim sen nası hepsini ifli çizdin chapter logolarını, kafama yapıştırmadın monitörü, "hade lennnn" diye, hala merak ederim)

Benim gözümden ve aklımdan geçenler budur.

Drey'in 7dx2010 Raporu

Emir 'Drey' Diril

Bu sene hiçbirşey yapamadım bari adam gibi bir parti raporu yazayım.

Her ne kadar 7DX partisinin bu seneki ayağına sadece cumartesi günü iştirak edebilmiş olsam da benim için yine de iki günlük bir demoscene macerası oldu. Şöyle ki:

Önceki haftasonu Hydrogen ile aramızda geçen telefon konuşmasında müsait ise partiden önceki cuma akşamı ona gelebileceğimi söyledim. Bu hem parti öncesi için güzel bir ısınma turu hem de henüz partiye birşey hazırlamamış olan ben ve çok geç başladığı için SID müziğini nasıl bitireceğini düşünen Hydro için gaz bir çalışma ortamı olacaktı.

Şekil 1.

24/12

Ve cuma günü geldi. İkimiz de feci derecede gazdık. Önceki partilerden konuşuluyor, müzikler dinleniyordu. Ben onu başladığı parçayı bitirme konusunda gazlarken o da beni pixel grafik alanındaki çekingenliğimi yenmem konusunda motive etmekten geri kalmıyordu (pixel grafik hastası bir insan olarak ezelden beri hep yapanlara hayranlıkla bakmış [bkz. ilk partim rehberi] ve bizzat denemeyi çok istemişimdir. Ama tabi Turbo, Spritus, Arcane, Hydrogen gibi babalardan tırstığım için elim hiç gitmedi bu vakte kadar).

Sonuç olarak o anki gazımız bizi bir iki saat kadar daha ürünlerimizin başında tutsa da kendisinin daha sonra lafa "Sen Iron Man oynamıştın di mi?" diye girmesiyle "üretken" ortamımıza ilk darbe inmiş oldu . Ben daha arabayı düz sürmeyi becerene kadar kendisinin beni bir kaç defa ezici bir şekilde yenmesi sonucu oyunumuz kısa sürdü tabi . Tam bu noktada belki de tekrar ürünlerimizin başına dönecekken bir ikinci darbe de Katakis ile geldi. Oyunun ne kadar zor olduğu ve benim gibi özellikle shoot'm up delisi bir adamı bile pes ettirdiği başlıkları üzerine dönen geyikler ve "Abi bir tur daha deneyelim bu sefer burayı geçicez." takıntısı gecenin sonunu getirdi zaten. Bu arada sahiden gemi eşşek kadar be kardeşim, nasıl geçsin o daracık yerlerden

Şunu da eklemeden geçmeyeyim. Cuma akşamı uğraştığımız ürünlerin dışında, yıllardır compolardaki 4/4'lük parçaların hakimiyetini yıkmak üzere Hydrogen'in 9/8 benim ise 7/8 bir parça hazırlama girişimimiz vardı partiden iki üç gün önce. Ama malesef ikimiz de bu sene müzik compoya ürün hazırlamak için makinelerimizn başına çok geç oturduk .

25/12

Gelelim cumartesi gününe. Sabah erkenden kalkılıp kahvaltı yapıldıktan sonra Hydrogen parti mekanına götürülecek gerekli donanımlarını toparladı. (bir an içine biri ekmek kutusu olmak üzere üç tane c64 koyduğu büyük çantayı alıp kaçmak istemedim desem yalan ama gel gör ki az sonra aynı arabayla parti mekanına gidecektik zaten). Ben zaten bir gece önceden kendisine gelirken hazırlığımı tamamlayıp geldiğim için toparlanma faslında yan gelip yattım. Hazırlık tamamlanınca ekipmanları arbaya yükleyip Moldibi Brothers' Mansion'ın yolunu tuttuk. Tabi ben arabada dört kişi olucaz o yüzden fazla yer kaplamayayım diye ön koltuğa koca sırt çantam ve gitarımla sığmaya çalışarak tüm zip dosyalarına meydan okudum. Ama daha sonra Endo'nun bize katılmayacağını bu yüzden üç kişi olacağımızı öğrendiğimde ve Datura'nın arkamdaki koltukta rahat rahat oturduğunu farkedince "Acaba sıkışma işini abarttım mı yau?" diye düşünmeden de edemedim .

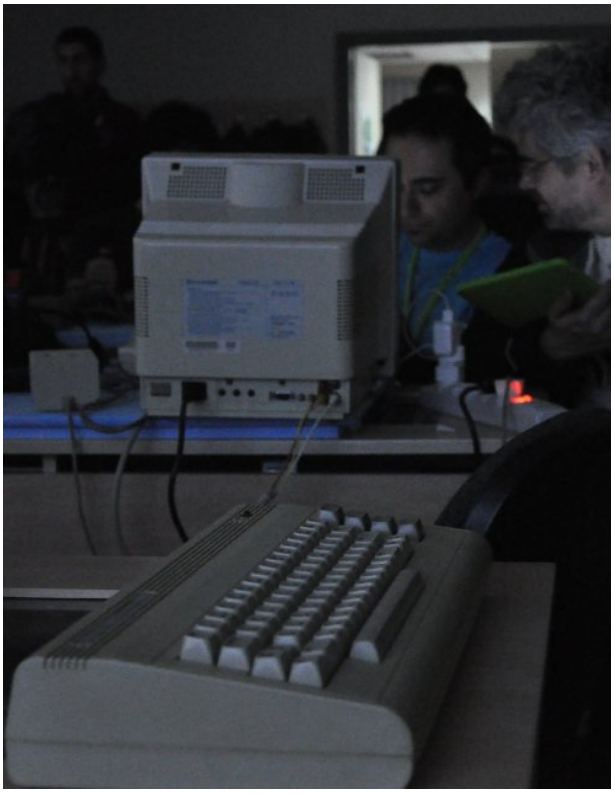
Parti mekanına varıldı, ekipmanlar içeri taşındı. Bu sene grubu olmayan RONIN bir scener olduğum ve pek ürün hazırlamakla da uğraşmayacağım için masa seçimi konsunda gayet özensiz davrandım. Gerçi partinin ilk birkaç saati pixellerimle baya uğraşsam da bir noktadan sonra tüm renkleri aynı görmeye başlamam sonucu grafiğim yine tamamlanamayarak bir başka bahara kaldı. Aslında bitirebilirdim ama pazar günü partide olamayacağımı bilmenin verdiği sıkıntı ve burukluk sanırım üretkenliğimin son iki damlasını da kuruttu.

Şekil 2.

Mekana yerleşmenin ve netbookumda bir iki pixel savuşturmanın ardından sıra gezinip tanıdıklarla merhabalaşmaya ve etrafta kim ne yapıyor, müzik yapan var mı, pixel çizen var mı diye bakmaya geldi (Aslında kod yazanları da oturup izleyesim var ama inanın bu konuda Demoscene'in yüz karası bir insan olarak hiçbirşey bilmiyor ve anlamıyorum. Öğrenmeyi de isterim ama gözümde dağ gibi büyüyor. Sonuçta bambaşka bir dünya). Hemen yeri belli olan organizatörlerden başladım. Girişte hemen sağda duvar dibindeki masada beyaz asus netoobk'u ile Endo, akabinde kürsüde multitasking olarak bir dünya şeyle haşır neşir olan ve beyninin multi-core desteğinden şüphe etmediğim adaşım Skate. Bu gezinme sırasında bir de ne göreyim, Norvax'ın laptopundan -Skype sağolsun- Nightlord partiye okyanusun öteki ucundan online olarak iştirak ediyor. Nasıl mutlu oldum onu görünce anlatamam. Tabi bunu takiben bu sene müzik hazırlayamadığımı söylerken de bir o kadar utandım, bir şekilde topu Hydro'ya atıp kaçtım ordan. Dedim ya zaten kapıdan girdiğimden beri "Ben bu sene eli boş geldim, partiye de sadece cumartesi günü katılabilicem." derken her seferinde canım sıkılıyordu, kaçıp bir köşede saklanma ihtiyacı hissediyordum

Daha sonra bir ara Hydrogen, Spritus ve Allamullax'ın yanına gittim. Allamullax'a "Müzik var mı bu sene?" diye sorduğumda Spritus ile bir demolarını olduğunu duyunca çok hoşuma gitti. Tabi Ayrıca Allamullax'tan yine o etnik öğelerinin olduğu parçalardan da beklerdim ama olsun. Demo yapmışlar daha ne. Bir ara Spritus şaka ile karışık olarak bana ciddi bi bakış atıp "Pixel falan işine girmişin, ekmeğimize mi göz diktin bakim?" dediğinde herhalde hemen arkasından yüzünde beliren o sıcak gülümsemeyi görmeseydim yüreğime inme inecekti .

Muhabbet sürerken Norvax geldi yanımıza. Kendisi ile ilk sohbetimiz bu partiye kısmet oldu. Açıkçası ben çok zevk aldım onunla sohbet etmekten ki akşam bir ara kapı önünde nefes almaya çıktığını görünce tüm yüzsüzlüğümle tekrar yanına gidip muhabbete girdim, hatta yanımda Chaotique de vardı. 2007'den beri TR'deki her demoscene partisine katılmama rağmen ilk defa bir akşam-sohbet-çemberi içinde buldum kendimi (bkz. ilk partim rehberi). Meğersem ne kadar zevkli ve gaza getiriciymiş. Tabi ki bunda sohbet ettiğiniz kişilerin de etkisi büyük. Norvax'a güzel sohbeti, şahsıma verdiği C64 gazı ve çenemi düşürebildiği için tekrar çok teşekkür ederim. Umarım onun da beni dinlerken başı fazla ağrımamıştır .



Şekil 3.

Geceden tekrar günün ortasına dönecek olursam benim için partinin en bomba olaylarından biri Datura'nın MSSIAH kartuşu diğeri ise LW3D'nin Atari Falcon'undaki Cubase programının ilk versiyonu idi.

MSSIAH'dan bahsedecek olursak, bu kartuş bence benim gibi hem retro hem de modern platformları bir potada eritmeye çalışan bir müzik tutkunu içim inanılmaz bir donanım. İçeriğindeki yazılımlar ve daha da önemlisi kartuşun üzerindeki MIDI IN girişi sayesinde C64'ünüzü herhangi bir ekstra modifikasyon yapmadan bir hardware synth haline getirebiliyorsunuz. Bu benim için şu demek; PC'den MSSIAH aracılığı ile C64'e midi komutları gönderilip bunlar SID'e çaldırılabilir ve tekrar ses çıkışı PC'deki host bir sequencer'da (Cubase,Pro Tools, Ableton live, vs..) yeni açılacak bir audio track'a atanabilir. Böylece pc ve C64'ünüzü senkronize bir şekilde müzik üretiminde kullanabilirsiniz. Bu da sizleri chip seslerini simüle eden yazılım synthlerden veya sadece rüyanızda görebileceğiniz Sidstation gibi donanımlara ulaşma çabasından kurtarır. Yeme de yanında yat yani, hem niye yiyosun canım kartuşu zaten .

Şekil 4.

Cubase'e gelince, LW3D'ye bu fırsat için tekrar teşekkür ederim. Sayesinde bir efsanenin çocukluğuna tanık oldum diyebilirim. Yanıma gelip "Ben de seni arıyordum, bak sana hoşuna gidecek birşey göstericem" dediğinde böyle bir süpriz beklemiyordum açıkçası. Hatta programın bu ilk versiyonunu biraz çözdükten sonra müzik yapma girişimim bile oldu ama sanırım sampleları içeren bazı module dosyaları olmadığı için ses alamadım. Olsun

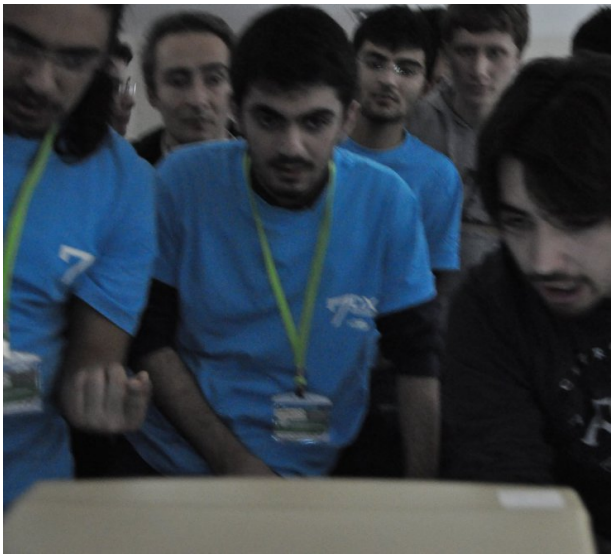
Akşam bir ara Impetigo'yu sırtında bass gitarı ile kapıdan girerken görünce çok mutlu oldum. Bu sefer benim de gitarım yanımdaydı, belki bir jam session yapma fırsatı yakalardık. Gel gör ki ikimiz de enstrümanlarımızın sesini duyabilmek için kulaklıklarımız dışında bir hoparlör vs getirmediğimiz için bu da başka bir partiye kaldı.

Seminerlerin hepsi çok güzeldi yine her zamanki gibi. Özellikle yukarıda bahsettiğim MSSIAH sonrası Datura'dan tekrar müzik dünyasına bir dönüş bekliyorum . Hydrogen'in SID müzik seminerinin de şahsım adına çok eğitici olduğunu söyleyebilirim. Özellikle C64'ün ilk sahneye çıktığı zamanki müzikler ile ilk olgun dönemde yapılan müzikler arasındaki o büyük fark gerçekten ağzımı açık bıraktı. Vay be!

Malesef açlığıma yenik düştüğüm için Nerdworking seminerini bölük pörçük izleyebildim. Sobee ise yine güzel bir tanıtımla Türkiye'deki oyun sektörü konusunda beni heyecanlandırdı. SüperCan her ne kadar "çocuk oyunu" olsa da gördüklerim gayet uluslararası standartlarda idi.

Sonuç:

Yine güzel bir demoscene partisi, sevdiğim ve hayran olduğum scenerları yine kanlı canlı görme fırsatı. Yine güzel seminerler. Ayrıca Compec'i de tebrik ederim, arı gibi çalıştılar. Organizasyon için ellerinden geleni yapan ve yaptıkları şeyi de bir o kadar sevdiklerine inandığım gençler. Ve tabi ki tüm organizatörlere de tekrar tekrar teşekkürler, yılmadan her sene artan bir kalite ile bize bu ortamı hazırladıkları için. Ellerinize sağlık.



Şekil 5.

Nightlord'un 7dx-2010 Raporu

Bilgem 'Nightlord' Çakır

Bu yıl parti oncesi gaza geliş yazımı yazamadım. Her yıl parti yaklaşırken ben de oluşan beklenti ve gaz sonucu "7dx partisi yaklaşıyor" konulu bir yazı yazarım. Bir süredir inaktif olan blogum bu yazıyla hareketlenir. Ardından parti gerçekleşir ve ben parti raporunu yazarım.

Bu yıl bir şekilde bu gaza gelmedim. Neden bilmiyorum. Yine partiyi bekledim tabii ki. Hatta yine Skype üzerinden 10 saat fark ile katılacağım için partiden önceki hafta izinli olmamdan faydalanarak, biyolojik saatimi bile kaydırdım. Her gece bir iki saat daha geç yatarak Cuma ve Cumartesi gecelerini sabahlayabilecek hale geldim. Fakat bu hazırlıklara rağmen ne bir ürün hazırlayabildim ne de parti öncesi blog ve forum yazıları yazabildim.

Oysa ki aklımda Patterns of Madness'ın devamı niteliğinde birşeyler yapmak fikri vardı. Bu sefer Direct2D kullanan bir demo yaparım diye düşünüyordum. Çeşitli ilginç 2D efektler hayal ediyordum zaman zaman.

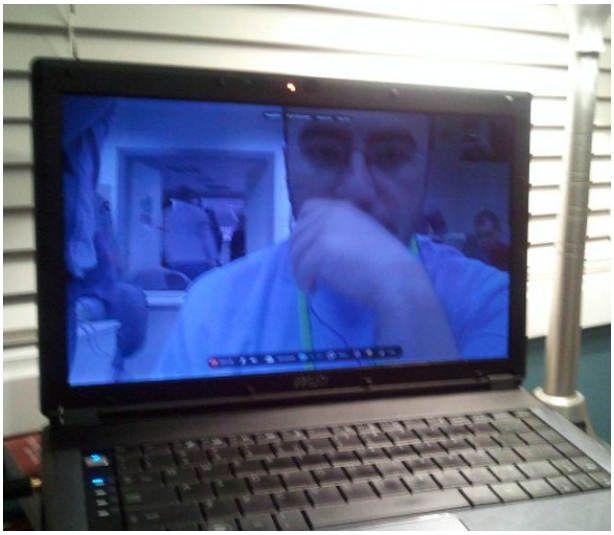
Ayrıca bir kaç ay önce PC'de müzik yapma olayını kurcalamaya başlamıştım. Güzel bir başlangıç seviye ses kartı (Presonus Audio Box) aldım. Gitarlarımı temizledim yeni teller taktım. Ses Kayıt teknikleri hakkında bir miktar araştırıp okudum. Küçük küçük müzikler ve kayıt denemeleri yapar oldum. Sonunda bir Drey haline gelemesem de yine bir şerefli ikincilik ya da üçüncülük kovalayabilirim diye düşünerek 7dx müzik compoya da bir mp3 sokmayı planladım.

Üstelik bütün bu planları uygulayacak vakit de vardı. Partiden önceki hafta dediğim gibi izinliydim.

Fakat bütün haftayı evde miskin miskin yatıp televizyon seyrederek geçirdim. Biraz okudum vs. En son Perşembe günü kendi kendime "Aman bu yıl sadece keyfini çıkarayım diğer ürünleri seyredeyim" falan dedim. Trapped bir ay kadar önce ilk trackleri kaydedip bıraktığım bir haldeydi. Az biraz mix edilmişti. Çok gaza gelirsem onu submit ederim diye düşündüm.

Sonunda Cumartesi geldi çattı. Ben kısa sürede Semih'le kontağa geçerek onun laptop'ı üzerinden Skype ile partiye bağlandım. Masama skype için laptop, bol kafein, su ve yiyecek hazırladım. Bir yandan desktop ile de sürekli facebook'a partiden twit spamlemek için hazırdım.

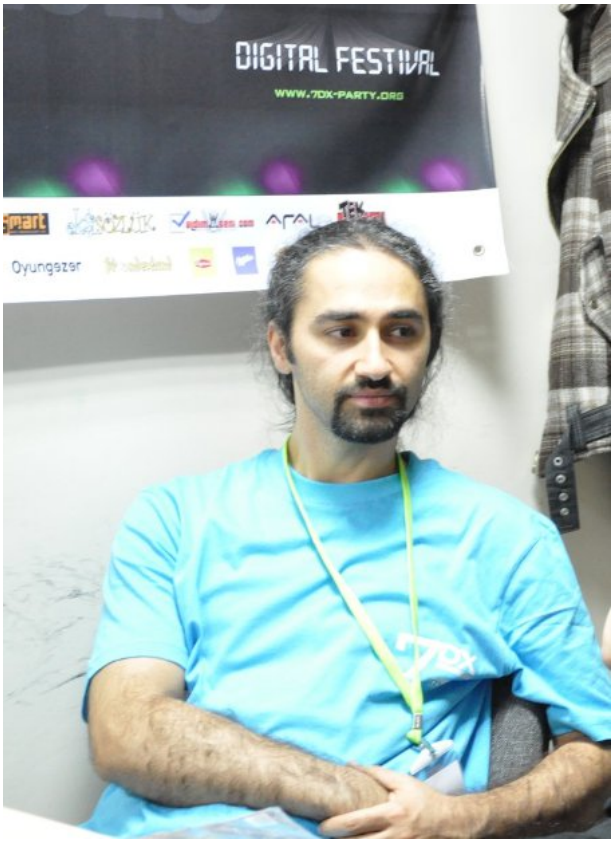
Önce Cumartesi günü Semih'in laptop'ına girip salonun sol orta bölümünde Return tayfasının hemen arkasına kuruldum. Önce Murqx ile biraz C++ muhabbeti yaptık, ardından Semih'le biraz re-senkronize olduk. Bu arada Aegis Skype'ın bulanık görüntüsünde beni Arcane zannedip "aa olm gelmiyo musun hadi gelsene" dedi. Aah ah keşke öyle ha deyince gelebilsem be abi



Şekil 1.

Sonra bana göre partinin en eğlenceli bölümlerinden biri gerçekleşti. Moldibi biraderler semineri. Benim şöyle bir dezavantajım var. Endo, Datura, Hydrogen ve Skate ile 5 yıl aynı grupta olmamıza rağmen tabi onlar 15 yıldır birarada oldukları için, bir ton hikayeyi zaten birbirlerine anlatmışlar geçmişte, o yüzden şimdi böyle bir araya geldiklerinde pek anlatmıyorlar. Halbuki adamlarda ne hikayeler var biliyorum. Bu seminer vesilesiyle en azında bi kısmını dinleyebildik. Duyduklarımdan da zaten yarıldım.

Bir de şunu belirtmeden geçemeyeceğim. Yani böyle bir adam tipi var. Bilgisayarları ile iki hafta oyun oynadıktan sonra biz de yapalım bundan diyen. Bunu kendiliğinden diyen. Web siteleri, dergiler, forumlar, partiler, yarışmalar olmadan. Yani düşünün takriben 10 - 12 yaşlarında iki çocuk. önlerinde bir bilgisayar duruyor. yanında bir kullanma klavuzu. O klavuzda 60 - 70 sayfalık BASIC artı grafik artı müzik bilgisi. Bunlara bakan bu iki çocuğun düşündüğü şey "biz de oyun yapalım. gereken herşey elimizde var. bütün oyunlar zaten basic ile yapılıyor olmalı, onu da bu kitaptan öğrenebiliyoruz. e tamam o zaman ne duruyoruz."



Şekil 2.

Üstelik eksik bilgileri yüzünden bir sürü emekleri boşa gidiyor. Basic'in falanca oyun projesi için yeterince hızlı olmadığını binlerce satır kod yazdıktan sonra öğreniyorlar mesela. Cevabın Simon's Basic olmadığını öğrenişleri de bir o kada boşa giden emek alıyor. Ve bu adamlar bütün bunlar olurken hiç bir noktada da "amaaaan lanet olsun" demiyorlar. Commodore dergisini bulup ilk introlarını, demolarını yaparken bile evlerinin dışında bir scene olduğunu bilmiyorlar. Yani scene'de şan kazanmak gibi bir dürtüleri de yok.

İşte bu adam prototipine ben hastayım. Bana göre en ulvi, en şanlı, en ayakta alkışlanması gereken adamlar bu prototip adamlar. Bu prototipten scene'de başkaları da var. Özellikle de Türkiye'de. Çünkü Türkiye bilgiye ulaşabilmek bakımından daha dezavantajlı durumda kalmış o dönemde. Bu da Türkiye'de o yıllarda bu işle ilgilenen insanların böyle tırnağıyla kazıyan adamlar olmasını gerektirmiş.

Bu yüzden ben Moldibi biraderler seminerini hayranlıkla izledim. Ayrıca ortamdaki genç nesil izleyicilere bu "kurcalama kültürünü", eline bir cihaz geçtiğinde bu cihazla ne tüketebilirim diye değil ne üretebilirim diye düşünme içgüdüsünü kanlı canlı göstermiş oldu Moldibi biraderler.

Şekil 3.

O seminerden sonra, kah demo gösterimleri kah mekana gelen yeni insanlarla yer yer sohbetler derken saat Cumartesi 18 civarlarında benim pilim bitmeye başladı (Burada saat sabah 8 olmuştu).

Bu arada kara bir haber aldım. Drey bu partide müzik yayınlamayacaktı. Yıkılmıştım. Partiyle ilgili en büyük beklentim, umudum suya düşmüştü. Tabii ki Drey'den yine aşmış bir (belki iki: bir senfonik, bir metal) parça beklerken rakam sıfır olunca, haaaaaaaaaaayııııııııııırrrrrrrr şeklinde yağmur yağan sokağa çıkıp göz yaşları içinde koşmaya başladım. En son asfaltta bir su birikintisinin içinde dizlerimin üstüne çöküp iki elimi yanlara açıp yüzümü yukarıya çevirdim. Ben birkez daha haaaayıııır diye bağırırken kamera yavaş yavaş yükseldi.

Tam olarak böyle olmadıysa da yaklaşık böyle birşeyler oldu. Drey suçlamaları kulak arkası edip ustaca topu hydrogen'e attı. Ha hahahaha bu arada o muhabbetimiz süperdi. Drey hocam valla özlettin kendini. Artı sen parça yap ki ben senin mix ve recording triklerine uzun uzun çalışabiliyim. heheh

O saatlerde Norvax ile yaptığımız teattiler sonucu, henüz hiç başka entry olmadığını öğrenince bu sefer başka bir telaşa kapıldım. Genelde 7dx partilerinin en zengin geçen composu olan mixed music componun başı dertte miydi. Sadece Return'den eski SID gurularından Slowhand'den bir SID geleceği haberini aldık. Bunun üzerine hiç olmazsa Slowhand'in yarışmada yalnız kalmaması için Trapped'i bitmemiş haliyle de olsa compoya göndermeye karar verdim. Trapped'in aslında yaklaşık 8-10 saatlik daha işi var. Tabii göndermeden önce bana göre en kulağa batan bir iki eksikliğini hızla gidermeye çalıştım. Bir saat kadar kurcalayıp Norvax'a gönderdim.

Bir yandan bu "bitmemiş ürünle compoya katılma" hali bana geçen seneki halimi hatırlattı. Çok kıl bir duygu olduğunu birkez daha hatırladım.

Ardından Cumartesi gecesi compolarına yetişebilmek için biraz uyumam gerektiğine karar verdim. böylece akşam 6'dan gece yarısına kadarki seminerleri kaçırdım. Keşke birileri o seminerleri videoya çekmiş olsa.

Cumartesi gece yarısı yeniden kalktım ve yeni Skype session'ını başlattım. Bağlandıktan biraz sonra compoların ertelendiğini öğrendim. Bu arada Norvax, Murqx ve adını bilmediğim üçüncü bir arkadaş Robo code için kasmaya başladılar. Skype ile aktif sohbet edemeyeceğimi anlamam üzerine içim gıdıklanmaya başladı. Bu esnada demo compoya, Return'den bir C64 demosu ve Spritus'tan bir PC demosu geleceğini duymuştum. Burada beni gaza getirmek için yeterli data vardı.

Öncelikle Spritus'un sonunda bir demoyla geri dönüyor olması, benim için halaylar ve havai fişeklerle kutlanması gereken bir sevinç kaynağı idi. Bana hep çok ilham vermiş ve az önce bahsettiğim "scener prototip"ine tam uyan bu şahane adam yapmış demosunu gelmiş. Skyoe ekranından çok net göremediğim için yorum yapmak için ürünün kendisinin parti sitesine yüklenmesini bekleyeceğim.

Şekil 4.

İkincisi, Joker'in introlar evresinden multi part demolar evresine geçmesinin benim için ne anlama geldiğini biraz açıklamam gerekiyor.

Joker çook çok enteresan bir fenomen benim için. Şimdi bu şahane adamın hikayesi bizim hep duyduğumuz scener hikayelerinden çok farklı. Scene'e girişi ve ilerleyişi de öyle. Hep duyduğumuz "c64 alıp kullanma klavuzundan başlayan küçük çocuk" değil Joker. Retro forumlarında takılırken "bi dakka ya ben öğrenmek istiyorum" deyip ortamdakilerin yazdığı bir avuç dokümanı okuyarak başladı. Ardından forumlarda küçük kod örnekleri ile sorular sorup ufak tefek elini "coder"lığın o pek de bilinmeyen sıkıcı ve zorlu tarafıyla kirletmeye başladı.

Bu noktanın çok iyi anlaşılması gerekiyor. Coder olmaya niyetlenen insanlarla coder olan insanlar arasındaki en önemli fark, bu yolda ilerlerken önlerine çıkan çeşitli epik anlarda verdikleri tepkilerdir. Bu epik anlar genelde dışarıdan gayet gösterişsiz görünürler. Öyle bir anla karşılaştığında kişi bir bilgisayarın karşısında oturmuş ekrana bakmakta ve ne olduğunu anlamamaktadır. Bu hiç de zevkli veya heyecanlı bir deneyim değildir.

Coder'lığın en zevkli anları başka zamanlardır. Tam "zone"da olduğunuz, takır takır kod yazdığınız, beyninizin yazdığınıza ful hakim olduğu zamanlar, veya yaptığınız şeyi çalışırken gördüğünüz zamanlar zevklidir. Bir odada bir ekrana boş boş bakıp "ya niye olmuyo, işte olması lazım" dediğiniz an zevkli değildir. Kabus gibidir. Başından kalkıp gidip yatmak istersiniz (ki bazen bunu yapmak da gerekir)

İşte bu epik anlarda, bir şekilde pes etmeyip devam edenler sonunda o anı aşarlar. Bu aştığınız anda bile gösteriş olmaz. En fazla bir oh dersiniz. Ama böyle bir spor filminde kahramanın son saniyede ağır çekimle gol atması falan gibi birşeyle uzaktan yakından alakası yoktur. Belki bu epik anları aşmayı zorlaştıran şeylerden biri de bu. Bir diğer problem de bu epik anları coderliğa yeni başladığınız dönemde çok sık yaşamanızdır. Bu da pek çok insanın bu işi başta bırakmasının bir sebebi.

Şekil 5.

İşte Joker bu epik coder anlarını, bizler gibi 10'lu yaşlarda orta okul'un sakinliği ve yaşamın anne-baba tarafından finanse edildiği konforlu yaşlarda değil, otuzlu yaşlarda, bir aile geçindiren iş güç sahibi bir insan olarak yaşadı. Bu benim kolay kolay akıl erdiremediğim birşey. Elbette her scener bu yaşımızda aile ve ev geçindiriyorken bir sürü zaman harcayıp bu işlerle uğraşıyoruz ama yıllaaaaaar önce oluşmuş bir temelimiz var. artık kod yazarken karşılaştığımız problemler, demin bahsettiğimiz epik anlar gibi değil. Daha çok yıllardır araba tamir eden bir adamın yeni bir arızaya bakması gibi. O yüzden Joker boyle coderlıkta ilk epik anları bir yetişkinin sorumlulukları ve iş yükü ile göğüsleyerek tamamen özgün birşey tecrübe ediyor. Ve ben buna çok çok büyük saygı duyuyorum.

Buna karşın Joker o epik anları son derece tutarlı bir tempo ile aşmaya devam ediyor. Bunun sonucu olarak çeşitli kilometre taşlarını bir bir aşıyor. Logo, scroll, müziği bir araya koyan ilk intro, ardından ikinci intro, ardından ilk parti release, ardından grup kurma, gruba eleman ekleme, ilk multi part demo, sırayla demo efektleri yapmaya başlama, demo efektlerine kendisinden özgün birşeyler eklemeye başlama (plazmaya zoom ekleme gibi). Bunlar bugüne kadar ulaştığı kilometre taşları. Ve bunlar yaklaşık 3-4 yıl içinde oldu. Bana "ya bir logo, bir scroll, bir müzik içeren bir demo yayınlamak istiyorum" dediği zamanlar henüz birkaç yıl önceydi.

Şimdi bu gözle Rising'e bakınız ve C64 koduna aşinaysanız, bu demonun neden beni bu kadar heyecanlandırdığını anlarsınız. Yaşasın Joker, yaşasın Return.

Bu arada Return demişken belirtmeden geçemeyeceğim bir nokta daha var. Adamlar aslanlar gibi partiye geldiler. masalarına kuruldular, stickerları ile ürünleriyle, ara sıra masalarına gelen gençlere verdikleri bilgilerle partide "oradaydılar". Aegis, Joker, Slowhand harikasınız. Keşke Slowhand'de bir partiye gelse de yüz yüze tanışsak. gerçekten çok isterim.

Return'ün bu partideki "katılımı" bana Demodojo 2007'yi hatırlattı. Umarım Return artık bundan sonraki partilerde hep olan bir "köşe taşı" olur. Hatta belki yabancı partilere beraber gider orada "Türk gruplar" olarak takılırız.

Eveeet bu uzun parantezi kapatırken, sanırım neden demo compoda Return ve Resıdent demoları olacağını duymanın beni gazladığını anlatmış oldum.

Şekil 6.

Bu gazın ve parti mekanındakilerin uyku veya robocode ile meşgul olmasının sonucu olarak Cumartesi (aslında Pazar) gecesi 01.00 civarı ben de demo compo için küçük birşeyler yapmayı denemeye karar verdim. Daha önceden hali hazırda bir 2D animasyon framework'um vardı. Bu frameworkü kullanarak hiç olmazsa tek ekranlık birşey yapayım, demo compoda en azından 2 yerine 3 katılım olması, beni gaza getiren diğer iki demoya karşı da daha anlamlı bir saygı duruşu olur diye düşündüm.

Bunun üzerine framework'e hızlıca bir göz atıp (aylardır koda bakmamıştım)herşeyi tekrar derleyip hazır hale geçmeye çalıştım. Hemen müzik çaldırmak için bass kütüphanesini indirdim ve help dosyalarını gözden geçirdim. yaklaşık bir saat içinde demo üzerinde çalışmaya başlayabilecek hale geldim. Bu arada müzik olarak da Ağustos ayında yaptığım Rain Flower'ı kullanmaya karar verdim.

Bunu takip eden yaklaşık 11 saat içinde demonun genel temasını, ve kullanabileceğim efekt fikirlerini oluşturup teker teker partları hazırladım. sonuçta mümkün olduğunca temiz görünümlü minimalistic bir "mood" demosu yapmaya çalıştım. Müzik yine kafamda pekçok efekti oluştururken rehber oldu.

Pazar saat 12.00 civarı ekranları bitirmiş ve ince(mtrak) senkronizasyonu yapmıştım. Sonuçtan da gayet memnundum. Benim hard core coderpron demolarımdan biri değildi sonuçta ama tam olarak istediğim görüntü ve mood'u da yakaladığını düşündüm. Demoyu ikinci bir makinede (skype yaptığım laptop) deneyip Norvax'a gönderdim. Compo makinesinde sorunsuz çalıştığı haberini de aldıktan sonra partinin artık en heyecanlı bölümüne hazırdım. Yine masadaki çay, su, yiyecek stoklarını tazeleyip compoları beklemeye başladım.

Bir süre sonra compolar başladı ve ben yine kalbim küt küt atarak heyecanla izlemeye başladım.

Bu compo heyecanı apayrı bir olay. Compolarda ürün yarıştırmak hiçbirşeye benzemiyor ve benim için hiç olağanlaşmıyor da. bu kaçıncı parti, kaçıncı müzik, kaçıncı demo hala 7d4'te İkitelli'deki parti mekanında merdivenlere oturup OMOT'u seyrederkenki kadar heyecanlanıyorum. Eğer ürünüm tam olarak bitmemişse büyük bir ekşilik oluyor içimde. Ama bir de ürün bitmiş ise resmen kalbim fırlayacak gibi oluyor. Tabi yanında bir ton da endişe. Ürün oynatılırken bir aksilik olacak mı? demo göçer mi? ortasında salonda birşey olur mu? yani ideal demo izleme deneyimi olan karanlık oda, kesintisiz bir şekilde baştan sona art arda demolar şeklinde olacak mı Nitekim bütün bu endişe kafada konuşurken de bir yandan mantığım "boşa endişeleniyorsun Bilgem. kac tane compoda yarıştın hangisinde birşey ters gitti?" şeklinde cevap veriyordu. Ama bu sefer şans benden yana değildi

Demo'yu oynatabilmek için sağolsun organizatorler canla başla uğraştılar. Önce ses çıkmayan makineyi değiştirdiler, ardından yeni makinedeki virus programı demonun ortasında notifikasyon yapınca yeniden demoyu oynattılar. Gerçekten ne kadar teşekkür etsem azdır. Bunların hiçbirini yapmak zorunda değillerdi. Sonuçta program akışını aksatmamak için demoyu atlayabilirlerdi. Bu yüzden organizasyon gerçekten sonsuz minnettarım.

Tabi sonunda demo oynadı ama ben bayağı bir demoralize oldum. Demonun zaten tek numarası atmosferiydi, o da etkisini büyük ölçüde yitirdi diye düşündüm.

Tabi bu compoların en sonundaki ruh hali. Arada ruh halinin tavan yaptığı çok önemli bir nokta var atlamamak gereken. O noktanın adı: ZOMCOOOOOOOO!

Zomco partinin ihtimalen en bomba ürününü patlattı. Öz Desert Dream adlı Wild kategori ürünleri ile ilgili birşey şimdilik yazmayayım. Daha sonra elbet daha detaylı konuşuruz. Ama abi o ne yaaaa

Benim gözümde 7d5'ten Esas Invaders'ı da geçerek, 7dx tarihinin bir numaralı Wild entry'si oldu bu ürün. Zomco'ya helal olsun diyorum.

Şimdi Zomco da benim için ayrı bir yere sahip. Domino zaten Türkiye'de Amiga camiasının en önde gelen adamlarından biri. Bu güzel adam ilk önce 7d8 partisine geldi. güzelce katıldı döndü (kardeşi de vardı sanırım). O sene sadece Pazar günü katılmışlardı sanırım. Ertesi yıl Yatuyu ile birlikte iki gün ful katılıp gece mekanda kalıp, ürünler yayınlayıp (hem de biri Amiga oyunu), partiden hemen sonra güzelce parti raporunu yayınlayarak tam üsturubuyla partiye katıldılar.

Şekil 7.

Buna dikkat çekmek istiyorum. Zomco'nun 7d9'daki varlığı bence harikaydı. Yani bu partilere baktığımız zaman hepimiz ne görüyoruz. Bir demoscene partisi bir çok insanın birşeyler yapması ile şekillenen ve zenginleşen bir olay. Bir konferans veya fuar gibi değil. O tarz olaylarda organizasyondan sorumlu insanlardan başka kimseye düşen bir iş yoktur. Sinemaya gidip film seyretmekten çok farklı değildir.

Ama demo partilerde katılımcılara aslında çok büyük iş düşer diye düşünüyorum ben. Organizasyona yardımın dışında, compoların seviyesini tanımlayan şey katılımcıların ürünleridir. Her scener üstüne düşenleri elinden geldiğince yaparsa toplam deneyim herkes için daha zenginleşir. Bu sorumluluklardan ilk aklıma gelenler; en başta partiye gelmek (mümkünse gece kalmak), partiye ürün getirmek, partiden sonra parti raporu yazmak, partide çıkan ürünleri yapan kişilere birkaç satır da olsa geri bildirim vermek vs... Bunların hepsi o kadar önemli ki... Hem partiyi var ediyorlar, hem partiyi tarihe kayıt düşüyorlar. Elbette kimse bunları yapmaya zorlanamaz ama herkesin böyle binbir zorlukla meydana çıkarılan bir organizasyonu, kendi çapında bu şekilde desteklemeye çalışmayı biraz sorumluluk olarak görmesi, yapabildiğini yapması, yapamadığı hakkında birazcık kötü hissetmesi, hepimizin severek parçası olduğu bu partilerin devamını sağlayacak.

İşte bu yüzden Zomco'nun 7d9 katılımı örnek katılımdı. Yukarıda saydığım herşeyi süper profesyönel bir şekilde yaptılar. O

yüzden bu yıl artarak katılacaklarını haber aldığımda çoktan gaza gelmiştim bile ben.

Ve bu gazı boşa çıkarmayıp partinin en bombastik ürününü yaptıkları için bir kez daha Zomco'yu saygıyla selamlıyor ve ekliyorum: ZOMCOOOOOOOOOOOOOO!!!!!!!!!

Sonunda compolar bitti. Artık yavaş yavaş partinin sonlarına gelmiştik. Bir süre sonra sonuçlar açıklanmaya başladı ve bir kez daha kalp küt küt moduna geçildi.

Aslında ben genellikle compo sonuçları açıklanırken kendi ürünüm oynarkenki kadar heyecanlanmıyorum. çünkü orada ters gidecek bisey yok. urün oynamış olay bitmiş. Yani organizatör sonucu duyururken kilitlenmeyecek mesela :P Ama bu sefer compo sonuçlarının yakın olacağını beklediğim için biraz daha heyecanlandım. Nitekim müzikte ben Slowhand'in kazanmasını bekliyordum. Tekrar ellerine sağlık abi.

Demo compoda da söylemem gereken çok hayati önemli birşey var:

PARADOX, PARTİYE GEL!!!

Bu süper önemli. bu kadar üretken ve çalışkan bir adamın, demolarında çok kolay düzeltilip iyileştirilebilecek 3-4 nokta yüzünden, ulaşabileceğinin çok altında sonuçlar alması beni çileden çıkarıyor Bir şekilde fiziksel olarak bizzat partiye gelmen gerekiyor. Tekrar söylüyorum.

PARADOX, PARTİYE GEL

Artık bu yazının sonuna geliyorum daha sonra başka ilaveler de yaparım. Ve gece oldu yatmam lazım. Şimdilik bu kadar. Herkese bol sevgi selam. Bilahare adet olduğu üzere bir kişi kişi greets şeysi de yapacağım.

LW3D'nin 7dx-2010 Raporu

Gökhan 'LW3D' Sönmez

Her sene olduğu gibi bu yıl ki demopartyi heycanla bekledik. Forumdan görülebileceği gibi hazırlıklarımızı elimizden geldiğince yapmaya çalıştık.

Her ne kadar demo partide tamir edeceğimi düşündüğüm PC Engine DUO konsolunu evde hala bulamamış olsam da, arkadaşlarıma götürmeyi planladığım malzemeleri hazırlayıp, ozkano'yu beklemeye başladım. Özkan'la partilere birlikte gitmek bizim için bir gelenek oldu.

Şekil 1.

Cumartesi biraz geç de olsa (iş meseleleri), parti mekanına vardık.

Parti mekanına adım atar atmaz, hemen bir iki arkadaşla selamlaşıp, birbirimize sarıldık (isimlerini bilmesenizde her partide bir arada olduğunuzdan çok yakın hissediyorsunuz kendinizi..)... Parti mekanına vardığımızda, seminer olduğu için ışıklar kapalıydı ve mekandaki diğer arkadaşları görmemiz kolay olmadı. Bizde bu zamanı, yanımızda getirdiğimiz cihazları ve özkanın getirdiği kasalarca retro malzemeyi mekana taşımakla değerlendirdik.

Girer girmez dikkatimizi önceki senelere nazaran masaların daha dolu olduğu çekti. Girişin sağ tarafındaki arka kısmı daha önceki seneler geç gelsek bile kapabilirken, bu sene bu kısım doluydu. Alcofribas bizden daha önce gelmiş, mekanın ön kısmında yer tutmuştu. Alp Yener ve Yatuyu bile (şehir dışından gelmelerine rağmen) bizden önce gelip yer ayırmışlardı. Bizim için yer teklifinde bulunsalarda, ozkano ve benim eşyaları yaymamız için en az 3-4 masa gerekiyordu. Zaten mekanın retro cihazların ağarlıklı olduğu kısımda mekanın sağ ön kısmıydı (ilker'in getirdiği Amstrad'lar sayesinde)

Dikkatimizi çeken ikinci şeyse, önceki yıllardan biraz farklı bir katılımcı profilinin mekanda olmasıydı. Yeni arkadaşların partiye ilgi göstermesi çok memnun edici bir gelişmeydi. Ama önceki yıllardan görmeye alıştığımız başta Vigo olmak üzere diğer arkadaşların olmamaları bizi üzdü. Oldukça kısıtlı ve az sayıda insanın bir araya geldiği böyle organizasyonlarda insan herkesi görmek istiyor. Bakkada yine yoktu Neyse ki DemoDojo'dan diğer arkadaşlar oradaydı...

Şekil 2.

Sevindirici olan bir diğer hususta commodore.gen.tr müdavimlerinin çok büyük ilgi göstermiş olmalarıydı. Zonguldak'tan kalkıp gelen Alp Yener, Yatuyu, Bursa'dan gelen Screen, partilerde görüşmeye alıştığımız Arda, Joker, tabacı, Ozkano, Alcofribas, commodorefan, wizardofwor, ilkergorkem, zen, Allen, GokhanOzkan, selimgokhunavci ve şu anda bir anda aklıma gelmeyip sonradan ekleyeceğim diğer arkadaşlar bir yandan ozkano'nun getirdiklerini karıştırıp, bir yandan hoşça sohbet ettik. Gelenler arasında sanırım en önemli katılımcıysa Zafer'di... (seni unutmadım Zafer...Nasıl unutabilirim ki ?)

Forum üyeliğinden çıkardığımız, Zafer partinin bence unutulmazı olarak zihinlerde yer etti. Commodore.gen.tr'nin her üyesiyle tanışıp, demo partide herkesle sohbet etti. Forumda yazdıklarından dolayı, kendisine çokça kızan birçok arkadaş, kendisini tanıyınca, fevri davranışını mazur görme yolunu tercih etti. Kendiside yazdıklarından yaptıklarından çok pişman olduğunu, DefLeppard'dan özür dilemek için yol aradığını ifade etti. Fotoğraflara bakınca ne kadar genç bir eski üyemiz olduğunu göreceksiniz... Parti mekanına getirdiği VirtualBoy çok ilgi çekti. Pillerinin kısa sürede bitmesi nedeniyle çoğu kişi VirtualBoy'u dene-

yemedi ama, CDTV, CD32, VirtualBoy retro cihaz dengesini olumlu yönde etkiledi..

Bu seneki DemoParti'nin bizim için önemli bir özelliğide Tolga Abacı arkadaşımızın seminer verecek olmasıydı. Tahminlerin oldukça ötesinde, çok derti toplu güzel bir tanıtımla bizlere Retro bilgisayarlar için modern depolama yöntemlerini ve projelerini anlattı. Tolga sadece anlatmakla kalmayıp UFE projesinin 3 örneğini bizlere (Bana, ozkano ve alcofribas'a) hediye etti. Bu zaten partinin bizim için en bomba yönüydü. Bu ürünün ilk örneklerine sahip olmak, çok mutluluk verici. Hem seminer hem bu güzel hediye için Tolga'ya çok teşekkür ederim.

Partilerin bizim için engüzel tarafı retro cihazlar hakkında saatlerce sohbet etmek... Tolga, Hydrogen, Arda, ozkano, commodorefan, Zen gecenin geç saatlerine, hatta sabaha kadar sohbet ettik. Arada Zafer'in gelip, " - ben demo yazmak istiyorum, hangi programda yazarım" sorularına cevap verdik. Tolga ve ozkano bir süre arızalı cihazları tamire koyuldular. Bir ara Joker'in C64'üne bir işlem yapıldı. Parti mekanına gelen osilaskop cihazının nasıl kullanıldığıyla ilgili kısa bir eğitim veridi. Arda Osilaskop'u bozmak için her düğmesine basıp, oynasada başarılı olamadı mekana getirdiğim Atari Falcon çok ilgi görmedi İçinde Cubase'den basşka birşey olmayınca, yeterince kurcalama şansımız olmadı. Arada Tolga'yla PES2011 oynadık ve PES'te önce dengeyi sağlasamda, sabahın ilk ışıklarıyla hezimete uğradım Hezimetten sonra, bit pazarına sefer düzenledik (ozkanı, tabacı, zen ve ben). Pek renkli bir pazar olmadı...Oldukça ıslaktı Ama yinede kısmetimde bir xbox varmış. Cihaz ilk denemede çalışmadı ama çalışmalarımız sürecek...

Şekil 3.

Eve gidince güzel bir kahvaltı yapıp hemen uykuya daldım...Rüyamda PES'te Tolga'yı şamarlıyor bir yandan Demo izliyordum

son olarak Skate, Endo, Hydrogen, Arcane, CaiSSon, mfk ve diğer demo parti müdavimleri ve commodore.gen.tr'den arkadaşlarla bir araya gelmekten, sabaha kadar retro cihazlardan konuşmaktan her zaman olduğu gibi yine çok büük keyif aldım. Organizasyonu düzenleyen ve katılan herkese çok çok teşekkürler ederim... Daha sonra unuttuklarımı ekleme ve hataları düzelme maksadıyla birşeyler eklemek üzere, benden şimdilik bu kadar..

Not:ozkano çektiği videoları paylaşabilirse, mekan ve commodore.gen.tr katılımcıları görülecektir...

Not2:Hades'in imzasında adaptorsuz Amiga gördüğümden yanımda bir adaptor getirmiştim. Hades gelmeyince, zafer adaptore el koydu...Kısmet...

Not3:mfk'dan UFE'nin üretimi için görüşüyorduk ama eposta üzerinden işler zor yürüyordu. Partide mfk ile görüşmek çok iyi oldu. UFE bir şekilde hayata geçecek Insallah...

C++ Kursu 4

Bilgem 'Nightlord' Çakır

İşaretçiler

Daha önce bir takım değişkenleri tanımlamaktan ve kullanmaktan bahsettik. Bu değişkenlerin hepsinin de bellekte durduğunu biliyoruz. İşte bir değişkenin bellekte durduğu yere o değişkenin “adresi” denir. Bu adres genelde platformun mimarisine bağlı büyüklükte bir sayıdır. Örneğin 32 bitlik bir sistemde bir değişkenin adresi de 32 bitlik bir sayıdır.

Bellek denilen şey art arda dizilmiş baytlardan oluşur. Bu baytlara istediğimiz bilgileri yazarız. Zaten siz bir değişken tanımladığınızda, o değişkenin tipine bağlı olarak bellekte bir grup bayt değişken için ayrılır. Örneğin çoğu 32 bitlik sistemde siz int tipinden bir değişken tanımladığınızda bellekte 4 baytlık bir alan ayrılır.

Bir değişkenin adresini almaya yarayan operatör & operatörüdür. Örneğin

```
int a = 1;
std::cout << &a;
```

dediğiniz zaman sisteminizin a için ayırdığı baytlardan ilkinin adresini ekrana yazdırmış olursunuz.

C++ bu tip değişken adresleri ile çeşitli işlemler yapabilmeniz için “işaretçileri” destekler. Bir işaretçi kendisi de bir değişkendir. İçerisinde başka bir değişkenin “adresini” taşır. Başka bir deyişle başka bir degişkene “işaret eder”. Bir nevi “şurada falanca tipte bir değişken var” der. Bunu biraz açalım.

İşaretçiler gösterdikleri değişkenin tipi ile bağlantılıdırlar örneğin

```
int a = 5;
int* pa = &a;
```

Burada önce a adında tamsayı tipinde bir değişken tanımladık ve o değişkeni hemen 5 ile yükledik. Ardından “int tipinde bir değişkene bakan”, pa adında bir işaretçi tanımladık. Ve ona a değişkeninin adresini yükledik. İkinci satır bu demektir.

İşaretçiler aracılığıyla onların gösterdikleri değişkenlere erişebiliriz. Bunun için *operatörünü aşağıdaki gibi kullanırız

```
std::cout<<*pa;
```

bu komut a’daki değeri yani 5’i yazdırır. Aynı şekilde

```
*pa = 10;
```

Diyerek a değişkeninin değerini değiştirebiliriz.

İşaretçiler ile diziler arasında da akrabalık vardır diyebiliriz. Bir dizi tanımladığımız zaman dizinin adını aynı zamanda dizinin ilk elemanına bakan bir işaretçi gibi kullanabiliriz. Örneğin

```
int dizi[3] = {1, 2, 3};
*dizi = 5;
std::cout<< dizi[0];
```

dediğimizde 5 sonucu yazdırılır. Ancak dizilerin işaretçilerden farkı başka bir int değişkeninin adresini dizi’ye atayamamanızdır (oysa dizi bir dizi değil de işaretçi olsaydı atayabilirdiniz)

Değişkenlerin nerelerde ne amaçla kullanıldığını merak ediyor olabilirsiniz. Bu kullanımları yeri geldikçe birlikte göreceğiz.

C++’da Bellek Yönetimine Giriş

İşaretçilerin kullanıldığı yerlerden biri dinamik bellek yönetimidir. Şu ana kadar bütün gördüğümüz değişken tanımlamalarında, derleyici programın derlenişi esnasında ihtiyaç duyulan bellek miktarını bilebilir ve bu belleğin ayrılmasını dolaylı olarak sağlar.

Oysa bazen programcı, bellek tasarrufu için ya da başka sebeplerle çalıştığı sistemden bellek isteyecek şekilde programı yazmaya ihtiyaç duyar. Bu durumlarda programımıza bazı durumlarda sistemden bir miktar bellek isteyen komutlar yazarız.

İşte “bana biraz bellek ver” diye konuştuğumuz şahsiyete aslında “C++ runtime” deniyor. Bu programınıza eklenen bir kod bütünü. İşletim sistemi ile sizin programınız arasındaki bazı bağlantıları yaparak, programınızın çalışması için zemin hazırlıyor. Runtime’ın yaptığı işler arasında programınızın ihtiyacı olan belleğin bir kısmını siz istedikçe size sağlamak da var.

Aynı şekilde kullanıp artık ihtiyacınız kalmayan belleği de yine sisteme iade etmek gerekir. Siz bir bellek parçasını sisteme iade ettiğinizde o bellek sonraki isteklerde size verilebilecek bir kaynak olarak müsait hale gelir.

Eğer hep sistemden bellek alır ve hiç iade etmezseniz, eninde sonunda sistemde bellek kalmaz ve programınızın başına kötü şeyler gelir. Bu şekilde iade edilmesi gereken belleği iade etmeyi unutmak C++ programlarında sık yapılan bir hata türüdür ve özel bir adı vardır: Bellek Sızdırma.

Şimdi sistemden nasıl bellek istediğimize bakalım:

```
int* pa = new int();
```

Sistemden bellek isterken new komutunu kullanırız. Bu esnada o bellekte saklayacağımız şeyin tipini de söyleriz. Ve bu istek bize bir işaretçi döndürür. Bu işaretçi yeni oluşturulan değişkenin adresini taşımaktadır. Örneğin yukarıda bir tamsayı saklayacak kadar belleği sistemden istiyoruz. Eğer runtime istediğimiz belleği bulabilir ise (bellek bitmemişse bulur) uygun büyüklükte bir bellek parçasını ayrırır ve o parçanın adresini döndürür.

Dönen şey bir adres olduğu için onu bir işaretçide saklamak zo-

rundayız. İşte bu işaretçilerin en çok kullanıldığı yerlerden biridir. Bu tamsayı değişkenini hep işaretçi üstünden kullanırız.

İşimiz bittikten sonra da bu belleği delete komutu ile sisteme iade ederiz

```
delete pa;
```

böylece bellek sızıntısı olmamış olur.

Benzer şekilde new ve delete kullanarak bir dizi saklayacak kadar bellek de sistemden isteyebiliriz. Üstelik new ile bu şekilde bellek isterken dizilerden farklı olarak değişken boylarda dizileri saklamaya yetecek kadar bellek alabiliriz. Bunu biraz açalım. Şu koda bakın:

```
std::cout<<"kac sinav";
int kacSinav;
std::cin>>kacSinav;
int sinavNotlari[kacSinav];
// malesef calismaz!!!
```

Burada dikkat ederseniz bir dizi oluşturmak istiyoruz. Ama dizinin kaç elemanı olacağını programı yazarken bilmiyoruz. Çünkü bu değer program çalışırken kullanıcı tarafından giriliyor. Kullanıcı belki 4 diyecek belki 500. Bu yüzden kacSınav değişkeni büyüklüğünde dizi tanımlamaya çalışıyoruz. Ancak bu kod derleyici de hata verir.

C++'da dizilerin bellek yönetimi derleyiciler tarafından ayarlanır. Derleyici bir dizi tanımı gördüğünde o dizi için gereken belleği alan kodu siz birşey yapmadan programınıza ekler. Derleyicinin bunu yapabilmesi için derleme anında gereken belleğin miktarını kestirebilmesi gerekir. Bu yüzden dile şu kural konmuştur. Dizilerin ebatları değişken olamaz, sabit olmak zorundadır.

Ancak new kullanarak derleme zamanı yerine çalışma zamanında bellek istemini yaparsak önümüz açılıyor. Bu durumlar için new komutunu şöyle kullanabiliriz.

```
Int* sinavNotlari = new int[kacSinav];
```

Tıpkı dizilerdeki gibi köşeli parantezleri kullanıyoruz. Bu ifadede parantezin içine değişken gelebilir. Çünkü istek program çalışırken yapılıyor. Programımız bu satıra geldiğinde kacSınav adet tamsayıyı saklayabilecek büyüklükte bir bellek parçasını sistemden ister. Sistem bulduğu belleğin adresini sınavNotları işaretçisine yükler.

İşaretçiler ve diziler akraba demiştik. Bu noktada [] indeksleme operatörünü tıpkı dizilerde olduğu gibi işaretçilerle de kullanabiliriz. Yani

```
s#navNotlar#[2] = 30;
```

gibi bir ifade geçerlidir.

New[] ile alınan belleği sisteme iade ederken delete komutu da biraz farklı yazılarak kullanılır:

```
delete [] sinavNotlari;
```

Bu şekilde ayrı bir delete kullanılmasının sebebi şu. Derleyici sınavNotları adlı int* işaretçisinin bir int'lik bir bölgeye mi yoksa n int'lik bir bölgeye mi baktığını bilmiyor. Burada delete[] kullandığımızda sisteme, "bu adresten başlayan daha önce new[] ile aldığım bütün belleği iade ediyorum" demiş oluyoruz.

Bu bilgileri kullanarak geçen sayıdaki örnek programımızı biraz daha geliştirelim. Bu sefer sinav notlarını iki boyutlu bir dizide tutmak yerine tek boyutlu "öğrenci sayısı x sınav sayısı" kadar tamsayı içeren bir bellek parçasında tutacağız. m'inci öğrencinin n'inci sınavına ulaşmak için de "(m x sınav sayısı) + n" formülü ile indeksleme yapacağız.

```
#include <iostream>

int main()
{
  // kac ogrenci
  std::cout<<"kac ogrenci?";
  int kacOgrenci;
  std::cin>>kacOgrenci;

  // kac sinav
  std::cout<<"kac sinav?";
  int kacSinav;
  std::cin>>kacSinav;

  // dongu ile notlari al
  #nt* sinavNotlari = new int[kacOgrenci * kacSinav];

  for(int o=0; o<kacOgrenci; ++o)
  {
    for(int i=0; i<kacSinav; ++i)
    {
      int index = o * kacSinav + i;
      std::cout << "sinav notu: ";
      std::cin >> sinavNotlari[index];
    }
  }

  // sonuclari hesapla ve goster
  int sinifToplam = 0;
  for(int o=0; o<10; ++o)
  {
    int ogrenciToplam = 0;
    for(int i=0; i<3; ++i)
    {
      int index = o * kacSinav + i;
      ogrenciToplam += sinavNotlari[index];
    }

    int ogrenciOrtalama = ogrenciToplam / kacSinav;
    std::cout<<"ogrenci "<< o
      <<" ortalamasi: " << ogrenciOrtalama
      << std::endl;

    sinifToplam += ogrenciOrtalama;
  }

  int sinifOrtalama = sinifToplam / kacOgrenci;
  std::cout << "Sinif ortalamasi: " << sinifOrtalama
    << std::endl;
}
```

Aslında asıl zevkli bölümlere yeni geliyoruz. Şimdi programcılıktaki en önemli kavramlardan birinden bahsetme zamanı

C++'da Fonksiyonlar

Fonksiyonlar bir programı dizayn ve organize ederken en faydalı araçlardan biridir. Fonksiyon genelde bir takım argümanı girdi olarak alıp, bir takım işlemler yaptıktan sonra bir sonucu geri döndüren kod bloklarıdır.

```
int kucukOlan(int a, int b)
{
  if (a < b)
  {
    return a;
  }
  else
  {
    return b;
  }
}
```

Burada bir fonksiyon görüyorsunuz. İlk satır fonksiyonumuzun dışarıdan bilinmesi gereken bilgilerini özetler.

int kucukOlan(int a, int b)

bu fonksiyon iki adet tamsayıyı girdi olarak alır. Bu iki sayıdan küçük olanı geri döndürür. İlk satırda sırasıyla fonksiyonun;

1. ne tipte bir değer döndürdüğü
2. adı
3. kaç tane ve ne tipte girdiler aldığı

ilan edilir. Programın geri kalanı sadece bu üç bilgiye dayanarak bu fonksiyonu çağırabilir. Bu üç bilginin bütününe fonksiyonun "imzası" adı verilir.

Kursun ilk bölümünden hatırlayacağınız return komutu değer döndüren fonksiyonlarda ikinci bir rol daha üstlenir. Return'den sonra gelen değer (sabit veya değişken) fonksiyondan döndürülen değer olur.

Tanımlanmış bir fonksiyonu programınızın geri kalanında adıyla çağırabilirsiniz.

```
int enDusukNot;
enDusukNot = kucukOlan(aliNot, veliNot);
...
```

Burada program akışı ikinci satırdan fonksiyonun ilk komutuna atlar. Fonksiyona geçirilen argümanlar aliNot ve veliNot, fonksiyonun içinde a ve b değerlerine eşitlenir. Ve fonksiyon a ile b'yi karşılaştırıp küçük olanını return komutu ile döndürür. Program akışı bu noktada tekrar fonksiyonu çağıran satıra döner. Fonksiyondan dönen değer (yani aliNot ve veliNot arasında küçük olan) enDusukNot adlı değişkene atanır ve program akışı bir sonraki satırdan devam eder.

Programın ilerisinde başka bir yerde aynı fonksiyonu başka değerlerle yine çağırabiliriz

```
enAzHarcl#k = kucukOlan(hasanHarcl#k, ayseHarcl#k);
```

Bu sefer tahmin edeceğiniz gibi aynı fonksiyona tekrar atlanır ve farklı iki argüman değeri a ve b'ye geçirilir. Fonksiyon bu sefer bu yeni iki değerin daha küçük olanını döndürür.

Dikkat ederseniz fonksiyonlar tekrar tekrar değişik girdilerle kullanılıp bize istediğimiz işi yapan faydalı araçlara dönüşürler. Bir nevi programın geri kalanına bir "hizmet sağlarlar". Bu yüzden çoğu zaman bir fonksiyonu çağıran başka lokasyonlara fonksiyonun "müşterisi" denir. Bu hizmet sağlama ve müşteri kavramlarını başka yerlerde de göreceğiz.

Şu ana kadar programlarımızda kullandığımız main bloğu da aslında bir fonksiyondur. Adı main olan bir fonksiyon. Fonksiyonlar bazen hiç argüman almayabilir ve hiçbir değer döndürmeyebilir. İşte kullandığımız main de öyle bir fonksiyon. Değer almayan bir fonksiyon tanımlarken fonksiyonun imzasında parantezlerin arası boş bırakılır. Değer döndürmeyen bir fonksiyon tanımlanırken de imzada dönüş değeri olarak "void" ifadesi kullanılır. Bu yüzden örneklerde gördüğünüz ana kod bloğumuz

```
void main()
```

diye başlıyor.

Fonksiyonların birkaç önemli faydası vardır:

1. Programınızda birden fazla yerde kullandığınız bir grup komutu fonksiyon haline getirirseniz o komutları sadece bir kere fonksiyon tanımında yazıp değişik yerlerde tekrarlamak zorunda kalmazsınız. Bu programınızın daha küçülmesini sağlar, ayrıca eğer o fonksiyonu oluşturan komut satırlarında bir hata yapmış ve sonra o satırları pekçok yerde tekrarlamışsanız (yani fonksiyon yerine her lazım olan yerde satırları copy/paste yapmışsanız) bu hatayı sonradan farkedip düzeltmek istediğinizde o copy/paste yapılan her yere gidip düzeltmeyi yapmanız gerekir. Oysa fonksiyon kullanırsanız hatayı bir yerde "fonksiyonun içinde" düzeltebilirsiniz ve bütün müşteriler bu düzeltmeden faydalanır.
2. Programınızın daha organize ve okunabilir olmasını sağlarlar. Biz şu ana kadar örneklerimizde herşeyi main fonksiyonu içinde yazdık. Bu çok küçük programlar için problem olmasa da zamanla okunmaz bir hal alır. Kod satırlarını alt fonksiyonlara bölerek organize etmek program yazmayı ve okumayı kolaylaştırır.
3. Programın değişik bölümlerinin sadece bazı detaylarla ilgilenip o detayların dışındaki konularla ilgilenmemesini sağlarlar. Bu aslında bütün bilgisayar biliminde çok önemli bir kavramdır ve buna "Soyutlama" denir. Mesela yukarıdaki örneğimizde küçük olan fonksiyonu, programın geri kalanını, "iki sayıdan küçük olanı bulma" probleminden soyutluyor. Bu şu anlama gelir. Programın geri kalanını yazarken iki sayıdan küçük olanı bulma ihtiyacınız doğarsa bunu nasıl yapacağınızı düşünmek veya yazmak zorunda değilsiniz. O problem kucukOlan fonksiyonunda bir kere çözüldü ve tekrar tekrar kullanılabilir. Bu, o fonksiyonu çağırdığınız yerdeki kod satırlarının daha sade ve odaklı olmasını da sağlar. Bunu daha ileride daha net göreceksiniz.

Biz şu an fonksiyonlar üzerinde çok durmayacağız. Çünkü asl-

ında C++ ile nesne yönelimli programlar yazarken fonksiyonlardan daha çok onlara çok benzeyen “sınıf metodlarını” kullanacağız. Bu konu da gelecek sayıda...

SDL Kursu 1

Emirhan (Ragnor) Bayyurt

SDL Nedir?

SDL, Simple DirectMedia Layer yani basit direkmedya katmanıdır. Biraz motamot bir çeviri oldu. Anlaşılır bir ifade ile SDL ses, klavye, mouse, joystick, 3d donanımı ve 2d grafik çizimi için hazırlanmış platform bağımsız bir çokluortam (multimedia) kütüphanesidir. Mpeg oynatıcılarda, Emulatorlerde, birçok popüler oyunda ve Linux'a port edilen birçok windows oyununda kullanılmaktadır.

SDL Linux, Windows, BeOS, MacOS Classic, MacOS X, FreeBSD, OpenBSD, BSD/OS, Solaris, IRIX, ve QNX ortamlarını destekler. Ayrıca resmi olmasada kod açık olduğu için gönüllü çalışmalar sonucunda Windows CE, AmigaOS, Dreamcast, Atari, NetBSD, AIX, OSF/Tru64, RISC OS, ve SymbianOS üzerinde de çalışması sağlanmıştır.

SDL C dili ile yazılmış olmasına rağmen C++ ise doğal hali ile çalışmakta ve Ada, Eiffel, Java, Lua, ML, Perl, PHP, Pike, Python, ve Ruby gibi birçok dil içinde uyarlamaları bulunmaktadır.

Neden SDL?

Çünkü SDL öğrenmesi ve kullanması kolay ama karşılığında çok güçlü bir kütüphanedir. Grafik konusunda OpenGL ile beraber çok kolay bir şekilde çalışabilmesine ek olarak eksik kaldığı her alan içinde SDL'i destekleyen ek kütüphaneler bulunmaktadır. Ses ve müzik için SDL_mixer, resimler için SDL_image, true type font desteği için SDL_ttf, ek çizim komutları için SDL_gfx ve network için SDL_net bunların en bilinen, en çok kullanılanlarındandır.

Bütün bunların yanında SDL platformdan bağımsızdır. Yazdığınız kod bir başka platformda neredeyse hiçbir değişikliğe gerek duymadan aynen çalışabilmektedir.

SDL Kurulum

Bu konu aslında basit olmasına rağmen yeni başlayanları biraz zorlayabilir. Yinede elimden geldiğince her platform'da SDL'i nasıl kurup kullanmaya hazır hale getireceğimizi yazmaya çalışacağım.

Windows

Windows ortamında en yaygın olarak MS Visual Studio, Borland C++ Builder ve Dev-C++ IDE'leri kullanılmaktadır. Ama ben daha önce hiç C++ Builder kullanmadığım ve internettede C++ Builder için yazılmış bir kurulum yazısı bulamadığım için size tahmini bilgi verebileceğim, gerisi size kalmış. Yinede pek sorun yaşayacağınızı sanmıyorum.

Visual Studio

İlk olarak MS'in IDE'si Visual Studio ile başlayalım. Ben Dev-C++ kullandığım için bu IDE hakkında pek bilgim yok ama internette yayınlanmış ingilizce derslerden topladığım bilgi ile yardımcı olmaya çalışacağım.

1. http://www.libsdl.org/download-1.2.php adresinden download bölümünden, Development Libraries başlığı altındaki Visual C++ için olan dosyayı (SDL-devel-1.2.8-VC6.zip gibi bir adı olması lazım) indirin.

2. Sıkıştırılmış dosyayı açın. Zip dosyasının içinden çıkan dosyalar arasında iki tane önemli klasör var. Bunlar "include" ve "lib" klasörleri. "lib" klasörünün içinden çıkanları C:\Program Files\Microsoft Visual Studio\VC98\Lib (büyük ihtimalle sizin bilgisayarınızdada aynı klasördür ama eğer MS VC++'ı kurduğunuz yer farklı ise bu yoluda ona göre değiştirin.) klasörü altına kopyalayın. "include" klasörü içindeki dosyalarıda C:\Program Files\Microsoft Visual Studio\VC98\Include\SDL (diğeri ile aynı şekilde farklı bir klasör olabilir sizin ki, artık duruma göre değişikliği siz yaparsınız. Birde include'un altında SDL klasörü yok onu sizin yaratmanız gerek.) klasörü altına kopyalayın.

3. Şimdi Visual C++'ı çalıştırın ve yeni bir proje yaratın. "Win32 Application" seçeneğini ve "empty project" seçeneğini seçin. File menüsündeki New seçeneği ile yeni bir c++ kaynak dosyası (c++ source file) yaratın ve adını main.cpp koyun.

4. Daha sonra proje ayarları (project settings) (project->settings menüsü yolu ile) bölümüne gidin. LINK tabına tıklayın listelenmiş diğer lib dosyalarının altına sdl.lib ve sdlmain.lib dosyalarını ekleyin.

5. Ardından yine proje ayarları bölümünde C/C++ tabına tıklayın. Drop-down menüden "Code Generation" seçeneğini seçin. Ardından da 'Use run-time library' drop-down menüsünden multithreaded DLL' seçeneğini seçin.

6. Son birşey daha. Sıkıştırılmış dosyanın içinde son bir önemli dosya bulunmakta. SDL.DLL dosyası. Bu dosyayı alıp ister işletim sisteminizin system klasörüne (win 95, 98, ME için c:\windows\system klasörü ya da Windows NT, 2000 and XP için c:\windows\system32 klasörü), ister uygulamanızın exe dosyasının çalışacağı klasöre kopyalayın. Eğer DLL dosyasını system klasörüne kopyalarsanız yapacağınız her sdl programı için dll dosyasını baştan baştan kopyalamanıza gerek kalmaz. Aksi takdirde her programın klasörüne koymanız gerekir. Ayrıca programlarınızı dağıtırken de bu dll dosyasını programınız ile vermeniz gerekiyor. Aksi halde yaptığınız program başka bilgisayarlarda çalışmaz.

MS Visual Studio ile işimiz bu kadar bütün bu adımları gerçekleştirdikten sonra MS VC SDL ile program geliştirmeye hazır hale geliyor.

Dev-Cpp

Dev-C++ için SDL yüklemenin iki yolu var. Birincisi http://www.devpaks.org sitesine gidip sdl için gerekli devpak dosyasını indirmek ve Dev-C++ içerisindeki Package Manager ile bu devpak dosyasını yüklemek. Bu sayede hem sdl kütüphanesi sorunsuzca bilgisayarınıza yüklenecek hem de elinizin altında hızla program yazmaya başlamanız için hazır bir sdl kodu olacak.

Ama ne yazık ki hazır sdl kodu içerisinde birkaç hata bulunmakta yine de bunları düzeltmek oldukça kolay.

Diğer yol ise yukarıdaki gibi geliştirme paketini indirip her dosyayı yerine kurmak. Bu yolu izlerken yapmamız gerekenler;

1. http://www.libsdl.org/download-1.2.php adresinden download bölümünden Development Libraries başlığı altındaki MinGW32 için olan dosyayı (SDL-devel-1.2.8-mingw32.tar.gz gibi bir adı olması lazım) indirin.

2. Sıkıştırılmış dosyayı açın. Zip dosyasının içinden çıkan dosyalar arasında iki tane önemli klasör var. Bunlar "include" ve "lib" klasörleri. "lib" klasörünün içinden çıkanları C:\Dev-Cpp\lib klasörü altına kopyalayın. "include" klasörü içindeki dosyalarıda C:\Dev-Cpp\include\SDL (include'un altında SDL klasörü yok onu sizin yaratmanız gerek.) klasörü altına kopyalayın.

3. Sıkıştırılmış dosyanın içinde son bir önemli dosya bulunmakta. SDL.DLL dosyası. Bu dosyayı alıp ister işletim sisteminizin system klasörüne (win 95, 98, ME için c:\windows\system klasörü ya da windows NT, 2000 and XP için c:\windows\system32 klasörü), ister uygulamanızın exe dosyasının çalışacağı klasöre kopyalayın. Eğer DLL dosyasını system klasörüne kopyalarsanız yapacağınız her sdl programı için dll dosyasını baştan baştan kopyalamanıza gerek kalmaz. Ayrıca SDL.DLL dosyasını c:\Dev-Cpp\dll klasörünün altına da kopyalayın.

4. Dev-Cpp'ı çalıştırın. New Project butonu ile yeni bir proje açın. Eğer Devpak'ı yüklediyseniz proje şablonları arasında Multimedia tabı altında SDL için hazır bir şablon hazır bulunmakta. Verdiğim dosya ile bu şablondaki dosyayı değiştirdiyseniz bu şablon sorunsuzca çalışacaktır. Bu şablon üzerinden dilediğinizce SDL uygulamalarınızı geliştirebilirsiniz. Ama eğer Devpak ile değilde SDL'in sitesinden indirdiğiniz sıkıştırılmış dosyadan kurduysanız proje ayarlarını elle yapmalısınız. İlk olarak yeni bir proje açın (New Project). Ardından Project Browser'da çıkan projenizin adına sağ tuşla tıklayın ve Project Options'ı seçin. Burda ilk olarak Type bölümünden Win32 GUI seçeneğini seçin. Eğer Win32 Console'u seçerseniz SDL programınız her çalıştığında önce dos penceresi açılır. Ardından programınız çalışır. Devamında aynı pencerede Parameters tabını açın. Compiler başlığı altına -I"<INCLUDE>\SDL" -Dmain=SDL_main parametrelerini girin. Linker başlığı altına ise -lmingw32 -lSDLmain -lSDL parametrelerini girin.

Ayarlamalar bu kadar. İsterseniz şimdi projenizi kaydedin ve SDL projesine başlayacakken ayarları hazır olduğundan bu proje üzerinden başlayın. Geriye bir tek kod yazmanız kaldı. O da sonraki bölümlerde anlatılacak.

Mingw32

Dev-Cpp'ta kendi içerisinde Mingw32 derleyicisi kullandığı için aşağı yukarı aynı ayarlar geçerli. Uzun uzadıya anlatmayacağım. Yukarıdaki açıklamalar yardımı ile sorunsuzca ayarlamaları yapabilirsiniz diye düşünüyorum.

Linux

Linux'ta SDL yüklemenin en kolay yolu kullandığınız dağıtımın paket yöneticisinden SDL paketini seçip indirmenizdir. Bir diğer yolda www.libsdl.org adresinden kaynak kodlarını indirip sırayla "./configure", "make" ve "make install" komutlarını vermeniz yeterlidir. Kodunuzu SDL ile derlemek içinde kodun içine '#include "SDL.h"' satırını eklemeniz ve derleyici parametrelerine de '-I"usr/include/SDL"' ve '-lSDL' parametrelerini eklemeniz gerekemekte.

SDL'e giriş

SDL öğrenmesi oldukça kolay bir kütüphanedir. Bu nedenle hızla kod yazmaya girişeceğim. Kısa sürede öğreneceksiniz zaten. SDL, 8 alt sistemin bileşiminden oluşmaktadır. Bunlar Ses(Audio), CDROM, Olay yönetimi(Event Handling), Dosya G/Ç(File I/O), Joystick yönetimi(Joystick Handling), Çoklu Görev(Threading), Zamanlayıcı(Timers) ve Grafik(Video)' dur. Bu alt sistemleri kullanmak için ilk önce bu sistemleri çalıştırmanız gerekir. Bunun için iki komut bulunmaktadır. Bunlar SDL_Init ve SDL_InitSubSystem dir. SDL_Init bütün SDL kodlarından önce çalıştırılmalıdır. Bu komut SDL sistemini çalıştırmaya başlar. SDL_InitSubSystem ise çalışma anında istediğiniz alt sistemin çalışmasını sağlar.

Kullanımları şu şekildedir:

```
SDL_Init ( SDL_INIT_VIDEO );
```

Bu komutla SDL programı çalıştırılır ve SDL'in video alt sistemi aktif hale getirilir.

```
SDL_InitSubSystem ( SDL_INIT_AUDIO | SDL_INIT_TIMER );
```

Bu komutla çalışmakta olan SDL programında ses ve zamanlayıcı alt sistemleri aktif hale getirildi. | işareti ile aynı anda birden fazla SDL alt sistemini seçebiliriz. Bu yönetmi SDL_Init komutu ile de kullanabiliriz.

Alt sistem bayrakları(flag) listesi:

1. SDL_INIT_TIMER - Zamanlayıcı -
2. SDL_INIT_AUDIO - ses -
3. SDL_INIT_VIDEO - grafik -
4. SDL_INIT_CDROM - cdrom -

5. SDL_INIT_JOYSTICK - joystick -
6. SDL_INIT_EVERYTHING - Bütün sistemleri aktif hale getirir -
7. SDL_INIT_NOPARACHUTE - SDL'in hata sinyallerini yakalamasını önler -
8. SDL_INIT_EVENTTHREAD - çok görevlilik -

Alt sistemleri çalıştırmayı öğrendik ama ya çalışan sistemleri kapatmayı? Şimdi de çalıştırdığımız SDL programını ve alt sistemleri nasıl kapatacağımızı öğreneceğiz.

İlk komutumuz SDL_Quit. Bu komut SDL_Init komutunun yaptığı işin tam tersini yapar ve başlattığınız SDL programını kapatır. Bu programı kullanırken herhangi bir argüman girmenize gerek yoktur. Kullanışı:

```
SDL_Quit();
```

şeklindedir. Bu komut ile hali hazırda çalışan bütün SDL alt sistemleri ve SDL programı kapanır. Diğer komutumuz ise SDL_QuitSubSystem. Bu komut ile çalışmakta olan istediğimiz alt sistemi kapatabiliriz. Kullanımı:

SDL_QuitSubSystem (SDL_INIT_TIMER);

şeklindedir. Örneğimizde zamanlayıcı alt sistemini kapattık ama SDL programımız çalışmaya devam etti.

Ayrıca SDL_WasInit fonksiyonu ile istediğiniz alt sistemin yüklü olup olmadığını kontrol edebilirsiniz. Bu fonksiyonunun yazılışı şöyledir:

```
if(SDL_WasInit(SDL_INIT_VIDEO)!=0)
    printf("Video alt sistemi yuklu.\n");
else
    printf("Video alt sistemi yuklu degil.\n");
```

Şimdi öğrendiklerimizle örnek bir SDL programı yazalım.

```
/* SDL header dosyas#. Butun SDL programlari
buna ihtiyac duyar */
#include "SDL.h"
#include <stdio.h>

int main() {
  printf("SDL programi ba#lat#l#yor.\n");

  /* SDL programi baslatilip Video ve Ses
  sistemleri aktif hale getiriliyor */
  if((SDL_Init(
        SDL_INIT_VIDEO
        |SDL_INIT_AUDIO)==-1)
        )
  {
    fprintf(
      stderr,
      "SDL programi baslatilamadi: %s.\n",
      SDL_GetError()
      );
    exit(-1);
  }

  fprintf(stdout,"SDL programi baslatilamadi.\n");

  fprintf(stdout,"SDL programi kapatiliyor.\n");

  /* SDL programi ve butun alt sistemleri
  kapatiliyor */
  SDL_Quit();

  fprintf(stdout,"Kapatiliyor....\n");

  exit(0);
}
```

SDL ve Grafik

Sırada grafik komutlarını kullanmayı öğrenmek var. İlk olarak yapmamız gereken şey video alt sistemini aktif hale getirmek. Ardından bir yüzey (surface) tanımlamalı ve SDL_SetVideoMode komutu ile bu yüzeyi kullanarak istediğimiz çözünürlükte bir pencere yaratmamız gerekir.

İlk olarak yüzey terimini açıklamak istiyorum. SDL'de ekrana çizdirmek istediğiniz bilgiler bir yüzeyde saklanıır. Bu yüzeyler aslında önceden tanımlanmış yapılardır (struct). Bir yüzeyi şöyle tanımlayabiliriz:

```
SDL_Surface *screen;
```

İçeriği ise şöyledir:

```
typedef struct SDL_Surface {
  Uint32 flags; /* Salt okunur */
  SDL_PixelFormat *format;/* Salt okunur */
  int w, h; /* Salt okunur */
  Uint16 pitch; /* Salt okunur */
  void *pixels; /* oku-yaz */

  /* kirpma bilgisi */
  SDL_Rect clip_rect; /* Salt okunur */

  /* Referans sayac# -- yuzey
  bosaltilirken kullanilir */
  int refcount; /* cogunlukla okunur */

  /* Bu yap# ayn# zamanda burda gösterilmeyen
  baz# özel alanlara sahiptir */
} SDL_Surface;
```

Gördüğünüz gibi SDL_Surface yapısı ile uğraşırken sadece pixels değişkenini kullanabilirsiniz. Bu değişkende yüzeyin her pixel'inin renk bilgisini taşıyor ve isterseniz her pixel'i teker teker değiştirebilirsiniz. Bu konuya ileride daha detaylı olarak bakacağız ama şimdi devam edelim.

Programın başında ekrana yansıtacağımız görüntüler için bir yüzey tanımlarız ve bu yüzey ile bir pencere açarız. Bunu SDL_SetVideoMode komutu ile yaparız. Kullanımı aşadığaki gibidir.

```
screen = SDL_SetVideoMode(
  640, 480, 8, SDL_SWSURFACE);
```

Bu komut ile screen yüzeyini ekrana yansıtacağımız ana yüzey olarak tanımlar ve 640'a 480 pixel çözünürlükte bir pencere yaratırız. Ekrandaki bit başına düşen pixel sayısı 8 olur. Ve

SDL_SWSURFACE bayrağı ile screen yüzeyine ait verilerin sistem belleğinde tutulması sağlanır. Burada kullanmak için birçok farklı bayrak bulunmaktadır. SDL_SetVideoMode komutu ile kullanabileceğimiz bayrakların listesi:

1. SDL_SWSURFACE -> yüzeye ait bilgilerin sistem belleğinde tutulmasını sağlar.
2. SDL_HWSURFACE -> yüzeye ait bilgilerin ekran kartının belleğinde tutulmasını sağlar.
3. SDL_ASYNCBLIT -> asenkron yüzey göstermeyi aktif hale getirir. Bu genellikle tek işlemcili makinalarda bit işlemeyi (blit - bit block transfer - bit bloğu değişimi) yavaşlatır ama SMP sistemlerde hız artışı sağlayabilir.
4. SDL_ANYFORMAT -> Normalde eğer video yüzeyi kullanılamayacak bir ekran derinliği (bpp) isterse SDL gölge bir yüzey ile bunu emule eder. SDL_ANYFORMAT bayrağı ile SDL'in bunu yapması engellenir ve SDL'in yüzeyin derinliğini umursamadan onu kullanması sağlanır.
5. SDL_HWPALETTE -> SDL'e ayrıcalıklı palet erişimi verir. Bu bayrak olmadan SDL_SetColors komutu ile istediğiniz renge herzaman ulaşamayabilirsiniz.
6. SDL_DOUBLEBUF -> Çifte tamponlamayı etkin hale getirir. Sadece SDL_HWSURFACE bayrağı ile beraber kullanılabilir. SDL_Flip komutu tamponların içeriğini değiştirir ve ekranı tazeler. Eğer çifte tamponlama etkinleştirilmemişse SDL_Flip bütün ekran üzerine SDL_UpdateRect komutu uygulanmış gibi davranır.
7. SDL_FULLSCREEN -> SDL tam ekran çalıştırmaya çalışıyor.
8. SDL_OPENGL -> OpenGL render ekranı yaratır. SDL_GL_SetAttribute komutu ile OpenGL ayarlamalarına başlamadan önce bu bayrağın etkinleştirilmesi gerekir.
9. SDL_OPENGLBLIT -> Üstteki gibidir ama aynı zamanda blitting (*yardım*) işlemlerine izin verir.
10. SDL_RESIZABLE -> Boyutlandırılabilir bir pencere yaratır. Pencere boyutları değiştirildiği zaman SDL_VIDEORESIZE olayı tetiklenir ve SDL_SetVideoMode yeni boyut ile tekrar çağırılabilir.
11. SDL_NOFRAME -> Mümkün ise çerçevesiz bir pencere yaratır. Tam ekran modu otomatik olarak bu bayrağı etkinleştirir.

Eğer istediğiniz ekran modunun uygun olup olmadığını öğrenmek istiyorsanız SDL_VideoModeOK fonksiyonunu kullanabilirsiniz. Yazılışı:

```
if (!SDL_VideoModeOK(
  640, 480, 16, SDL_HWSURFACE))
{
  printf("Ekran modu uygun degil.\n");
}
else
{
  printf("Ekran modu uygun.\n");
}
```

şeklindedir. Bunun dışında SDL_GetVideoInfo, SDL_GetVideoSurface, SDL_GetVideoDriverName ve SDL_ListModes gibi fonksiyonlar da bulunmakta ama şimdilik işin başında olduğunuz için işin başındaki sizlerin ihtiyaç duymadığı fonksiyonlar. Bunlara ileride detaylı değineceğiz ama şimdi sadece başka fonksiyonların da olduğunu bilin yeter.

Şu ana kadar öğrendiklerimizle bir SDL programını başlatıp SDL penceresini açabiliyoruz. Ama karşımızdaki simsiyah pencere oldukça sıkıcı değil mi? Hadi ortamı biraz renklendirelim. İlk olarak oldukça basit olduğu için bir BMP dosyayını okuyup ekrana yazdırmayı göstereceğim.

Bunun için ilk olarak okuyacağımız BMP dosyasının içeriğini saklayacağımız bir yüzey oluşturmalıyız. Yüzeyler SDL'de resim bilgisi saklanılacağı heryerde kullanılır.

```
SDL_Surface *image;
```

SDL'in kendi içerisinde BMP uzantılı dosyaları okuyup hafızaya alan hazır bir fonksiyonu bulunmakta. Adı SDL_LoadBMP . Kullanışı:

```
image=SDL_LoadBMP("c:\a.bmp");
```

şeklindedir. Bu sayede image adlı yüzeye a.bmp dosyasını yüklemiş bulunmaktayız. Şimdi sıra bu resmi ekrana çizdirmekte. Bunun içinse SDL_BlitSurface fonksiyonunu kullanacağız.

```
SDL_BlitSurface(image, NULL, screen, NULL);
```

Bu komut ile image yüzeyindeki resmi screen yüzeyine yani ekrandaki görüntünün saklanacağı yüzeye çizdiririz. NULL değer verilen parametrelerde çizdirilecek yüzeylerin boyutları ve koordinatları belirlenir. Eğer iki parametreye de NULL girersek image yüzeyinin tamamı screen yüzeyinin 0,0 noktasından başlayarak çizdirilir. Eğer resmin belirli bir kısmını çizdirmek istersek veya ekranda 0,0 noktasından başka bir noktaya koymak istersek ne yapacağız? SDL_Rect kullanarak.

SDL_Rect SDL içerisinde kare alan tanımlamak için kullanılan yapıdır. İçeriğinde sadece enine ve boyuna uzunluğu ile x,y düzlemlerindeki koordinatlarını saklayan değişkenler bulunur.

```
typedef struct{
  Sint16 x, y;
  Uint16 w, h;
} SDL_Rect;
```

x ve y koordinatları üst sol köşenin koordinatlarıdır. w ve h ise genişlik ve uzunluğudur. Oldukça basit ama niye böyle bir yapıya ihtiyacımız var diye düşünebilirsiniz. Bu yapıya ihtiyacımız var çünkü yüzeylerdeki resim alanları aslen dikdörtgen ve bunları kırpmak ya da belirli koordinatlara yerleştirmek isterseniz bu dikdörtgen yapısı oldukça kullanışlı oluyor. Yapmamız gereken şu:

```
SDL_Rect dikdortgen;
```

şeklinde tanımlamayılız. Eğer amacımız resmimini belirli bir koordinata koymak ise şu yönetmi kullanmalıyız:

```
SDL_Rect hedef;
hedef.x = x; // resmi koymak istedi#imiz
             // noktan#n x koordinat#
hedef.y = y; // resmi koymak istedi#imiz
             // noktan#n y koordinat#
SDL_BlitSurface(
  image, NULL, screen, &hedef);
```

Ama eğer amacımız resmin sadece bir bölümünü çizdirmek ise bu yönetmi kullanmalıyız:

```
SDL_Rect dortgen1,dortgen2;

dortgen1.x = x; // resmin ekran üzerine
                // yerle#tirilece#i x noktas#
dortgen1.y = y; // resmin ekran üzerine
                // yerle#tirilece#i y noktas#
dortgen1.w = w; // resmin ekran üzerine
                // çizilecek geni#li#i
dortgen1.h = h; // resmin ekran üzerine
                // çizilecek uzunlu#u

dortgen2.x = x2; // x düzlemindeki ba#lang#ç
                 // noktas#
dortgen2.y = y2; // y düzlemindeki ba#lang#ç
                 // noktas#

SDL_BlitSurface(
  image, &dortgen2, screen, &dortgen1);
```

Peki ya ekrandaki görüntünün bir kısmını bir yüzeye aktarmak istersek? Onu da bu yöntem ile yapabilirsiniz:

```
SDL_Rect dortgen1,dortgen2;

dortgen1.x = x1; // ekrandan kopyalanacak
                 // parçan#n sol üst
                 // noktas#n#n x koordinat#
dortgen1.y = y1; // ekrandan kopyalanacak
                 // parçan#n sol üst
                 // noktas#n#n y koordinat#
dortgen1.w = x2; // ekrandan kopyalanacak
                 // parçan#n sa# alt
                 // noktas#n#n x koordinat#
dortgen1.h = y2; // ekrandan kopyalanacak
                 // parçan#n sa# alt
                 // noktas#n#n y koordinat#

dortgen2.x = x1; // yukar#dakinin ayn#s#
dortgen2.y = y1; // yukar#dakinin ayn#s#

SDL_BlitSurface(screen, &dortgen2,
  temp, &dortgen1);
```

Bunun dışında SDL_Rect kullanarak ekrana bir dikdörtgen çizdirmenizde mümkün. Bunun için bir SDL_Rect tanımlıyorsunuz. Bu dörtgeni yerleştireceğiniz x ve y koordinatlarını, dörtgenin genişliği ile uzunluğunu ve rengini belirledikten sonra SDL_FillRect fonksiyonu ile ekrana istediğiniz koordinata istediğiniz boyutlarda ve istediğiniz renkte bir dörtgen çiziyor. Kod şöyle :

```
Uint32 renk; // dörtgenimizin renk de#eri
SDL_Rect dortgen;
dortgen.x = x; // dörtgeni ekran üzerinde
               // yerle#tirece#imiz x noktas#
dortgen.y = y; // dörtgeni ekran üzerinde
               // yerle#tirece#imiz y noktas#
dortgen.w = w; // dörtgenimizin geni#li#i
dortgen.h = h; // dörtgenimizin uzunlu#u
SDL_FillRect (screen, &dortgen, renk);
```

Oldukça kolay değil mi? Sanırım rengi nasıl belirttiğimizi merak ediyorsunuz. Ama şimdilik renk konularına girmeyeceğim ama ileride (az ilerde :)) detaylı olarak anlatacağım. Şimdi ise size ekran tazeleme fonksiyonlarını anlatacağım.

2D grafik programlamasında ekrana çizdireceğimiz görüntüleri önce çizdirmek sonra ekranı tazelemek ve sonra tekrar çizdirmek gerekir. Tazelemezsek ne olacağını basit bir örnek ile açıklayayım. Diyelim ki arkaplanda tam ekran çalışmakta olan bir penceredeki uygulamanız kilitlendi. Onun önündeki daha küçük bir pencerede çalışan uygulamanızın penceresini taşırsanır fark edeceğiniz üzere küçük pencerenin eski bulunduğu yerde görüntüsü (en azından görüntüsünün bir kısmı) hala durmakta. İşte ekrana çizim yaptıktan sonra ekranı tazelemezsek bu ve buna benzer bir sonuç alırız.

Peki ekranı nasıl temizleyeceğiz? Bunun iki yolu bulunmakta. Birincisi SDL_UpdateRect fonksiyonu ile.

```
SDL_UpdateRect(screen, 0, 0, image->w, image->h);
```

Ekrandaki görüntüyü sakladığımız screen yüzeyinin 0,0 koordinatından ekrana çizdireceğimiz image yüzeyinin genişliği ve yüksekliği boyunca uzanan alanı tazele komutudur bu. Bunun yerine bütün çizim işlemini bitirince ekranın tamamını tazeleyecek bir SDL_UpdateRect komutu daha kullanışlı olabilir. Şöyle ki :

```
SDL_UpdateRect(screen, 0, 0, 0, 0);
```

Ekranı tazelemek için kullanabileceğimiz bir diğer yöntem ise SDL_Flip fonksiyonudur. Bu fonksiyon sadece Video modu seçilirken çifte tamponlama (Double Buffering) bayrağı (SDL_DOUBLEBUF) seçilmiş ise kullanılabilir. Çünkü bu komut tamponların değişmesini sağlamak yolu ile ekranı tazeler. Eğer çifte tamponlama özelliğini kullanamıyorsanız bu komut yerine yukarıdaki bütün ekranı tazeleyen SDL_UpdateRect(screen,0,0,0,0); komutunu kullanın. Ama imkanınız varsa SDL_Flip'i seçmeye gayret edin. Kullanımı:

```
SDL_Flip(screen);
```

Bunun dışında birde SDL_UpdateRect fonksiyonunun SDL_UpdateRects adında birden fazla dörtgeni aynı anda tazeleyen farklı bir versiyonuda bulunmaktadır. Bu fonksiyonun kullanımı ise şöyledir:

```
SDL_UpdateRects(
  screen, dortgensayisi, *dortgenler);
```

Şimdi ise birkaç pixel fonksiyonu göreceğiz. Surface'lerin yapısını tanıtırken sadece pixels değişkeninin değiştirilebilir olduğu belirtilmişti. Bu değişkende yüzeyin pixellerinin renk bilgisi saklanmaktadır. Bu değişken dizisinin değerleri değiştirilebilir ve bu değişiklikler sayesinde ekrandaki pixellerin rengi değiştirilir. Bu işi yapan basit iki pixel fonksiyonu yazalım. Biri seçtiğimiz yüzeye istediğimiz renkte bir pixel yerleştirmeye diğeride seçtiğimiz yüzeydeki istediğimiz pixel'in renk değerini almamıza yarayacak.

```
Uint32 getpixel(
  SDL_Surface *surface, int x, int y)
{
  int bpp = surface->format->BytesPerPixel;
  /* p renk de#erini almak
  istedi#imiz pixel'in adresi */
  Uint8 *p = (Uint8 *)surface->pixels
    + y * surface->pitch + x * bpp;
  switch(bpp)
  {
    case 1:
      return *p;
    case 2:
      return *(Uint16 *)p;
    case 3:
      if(SDL_BYTEORDER == SDL_BIG_ENDIAN)
        return p[0] << 16 | p[1] << 8 | p[2];
      else
        return p[0] | p[1] << 8 | p[2] << 16;
    case 4:
      return *(Uint32 *)p;
    default:
      return 0; /* Bu sonuç ç#kmaz ama
                   ne olur ne olmaz. */
  }
}
```

Pixel dizisinde koordinatlar ilk 0,0 koordinatından başlar ilk elemanı ve her elemanda ilk olarak x değeri büyür. Bu büyüme pixel başına düşen bit kadar olur. Pixel başına 1 bit düşüyorsa her bit ayrı bir koordinattır. Ama 3 bit düşüyorsa 3 bitte bir koordinatları bir ileri gider. x koordinatı değeri limitine ulaştığında sıfırlanır ve y değeri bir artar.

```
Uint8 *p = (Uint8 *)surface->pixels
  + y * surface->pitch + x * bpp;
```

Satırında yaptığımız işlemde x değerini bpp değişkeni ile çarpıyoruz. bpp değişkeninde yüzeyin pixel başına düşen byte sayısını saklıyor. Yukarıda dizinin bir sonraki elemanının koordinat sisteminde ki bir sonraki noktanın rengini saklar demiştik ama eğer yüzeyde pixel başına düşen byte sayısı 1 ise doğrudur. Ama bazı durumlarda pixel başına düşen byte sayısı 2 veya 3'e çıkabilir. Mesela RGB renk paleti kullanıldığında her koordinat için ilk byte kırmızı renk değeri, ikincisi yeşil ve üçüncüsü mavi renk değeridir. RGBA renk paletinde ise ilk üçün RGB gibi dördüncüsü ise alfa değeridir. İşte bu yüzden x değerini bpp ile çarptık. Picth değişkeninde ise yüzeyin bir satırının uzunluğu tutuluyor. y değeri için bir demek x koordinatının limiti kadar gitmiş olmak ve bir nokta daha ileri gitmek demektir.

```
void putpixel(
  SDL_Surface *surface, int x, int y, Uint32 pixel)
{
  int bpp = surface->format->BytesPerPixel;
  /* p yerle#tirmek istedi#imiz pixel'in adresi */
  Uint8 *p = (Uint8 *)surface->pixels
    + y * surface->pitch + x * bpp;

  switch(bpp) {
    case 1:
      *p = pixel;
      break;
    case 2:
      *(Uint16 *)p = pixel;
      break;
    case 3:
      if(SDL_BYTEORDER == SDL_BIG_ENDIAN) {
        p[0] = (pixel >> 16) & 0xff;
        p[1] = (pixel >> 8) & 0xff;
        p[2] = pixel & 0xff;
      } else {
        p[0] = pixel & 0xff;
        p[1] = (pixel >> 8) & 0xff;
        p[2] = (pixel >> 16) & 0xff;
      }
      break;
    case 4:
      *(Uint32 *)p = pixel;
      break;
    }
}
```

getpixel fonksiyonuna oldukça benziyor. getpixel fonksiyonunda hedef pixel'in adresini bulup ordaki değerleri alıyorduk, burda ise yine hedef pixel'in adresini buluyoruz, ardından hedef pixel'in değerlerini istediğimiz renk değeri ile değiştiriyoruz. Dikkat edilecek nokta ise renk değerini değiştirirken pixel başına düşen bit sayısına göre hesaplama yapıyoruz. Oldukça basit ama kullanışlı.

Bu arada bu pixel fonksiyonlarını kullanırken veya bir yüzeyin pixel verilerilerine direk ulaşırken ilk olarak üzerinde çalışacağınız yüzeyi kilitlemelisiniz. Açıkcası dalgınlık ve merak ile kilitlemeden de çalıştırdığım oldu. Bunun nedeni ise bütün yüzeylerin kilitlenmeye ihtiyacı olmamasıdır. Neden diye sormayın bilmiyorum. Ama öyle.

Peki bunu nasıl anlayacaksınız? SDL_MUSTLOCK fonksiyonu ile. En iyisi size bunu yapan bir kod ile açıklamak.

```
if ( SDL_MUSTLOCK(screen) ) {
  if ( SDL_LockSurface(screen) < 0 ) {
    fprintf(
      stderr,
      "Yuzey kilitlenemiyor: %s\n",
      SDL_GetError()
      );
    return;
  }
}
```

Burda ilk olarak screen yüzeyini SDL_MUSTLOCK fonksiyonu ile kontrol eder ve kilitlenmeyi gerektirip gerektirmediğine bakarız. 0 değeri dönerse istediğiniz zaman istediğiniz pixele veri yazabilir, istediğiniz pixelden veri okuyabilirsiniz. Ama 0 verisi dönmezse bu yüzeyi kilitlemeniz gerekir. Böyle olunca da SDL_LockSurface fonksiyonu devreye girer ve parametre olarak girilen yüzey kilitlenir ama bir sorun olur da kilitlenemezse fprintf fonksiyonu ile stderr dosyasına "Yüzey kilitlenemiyor: Hata mesajı" şeklinde bir hata bildirimi yazdırırız.

Bu arada SDL 1.1.8 'den beri yüzey kilitlemek rekürsif, yani ardarda istediğiniz kadar kilit atabilirsiniz bir yüzeye ama bu gibi durumlardada attığınız her kilit için yüzeyi bir kez daha açmanız

gerekir.

Peki kilitlenen yüzeyi nasıl açacağız? SDL_UnLockSurface komutu ile. Kullanımı şöyledir:

```
SDL_UnlockSurface(screen);
```

Ama SDL_MUSTLOCK fonksiyonu ile beraber kullanmak istiyorsanız -ki tavsiye ederim- şöyle olacak:

```
if ( SDL_MUSTLOCK(screen) ) {
  SDL_UnlockSurface(screen);
}
```

Bu kadar basit.

İlk ders için bu kadarı yeterli. Şu noktaya kadar anlattıklarım ile SDL programları yazmaya başlayabilirsiniz.

Grafik Efektler için Matematik 3

Bilgem 'Nightlord' Çakır

Matrisler

Vektörler üzerinde çalışırken çok elemanlı vektörleri ve dönüşümleri kullandığımız için bu kavramları ifade ederken kullandığımız bir matematiksel araçtır matrisler. Mesela aşağıda bir matris görüyorsunuz

$$A = \begin{bmatrix} 2 & 3 & 2 \\ 4 & 7 & 1 \\ 6 & 0 & 3 \end{bmatrix}$$

Şekil 1.

Bu sudoku görünümlü şey aslında 3x3'lük bir matris. Matrisler m x n büyüklüğü ile sınıflandırılırlar. Burada m satır sayısı n ise sütun sayısıdır. Ve m ile n eşit olmak zorunda değildir. Örneğin;

$$B = \begin{bmatrix} 3 & 1.5 & 7 \\ 2 & 0 & 3.2 \end{bmatrix} \quad (2 \times 3 \text{ matris})$$

$$C = \begin{bmatrix} 2 \\ 6 \\ 1 \end{bmatrix} \quad (3 \times 1 \text{ matris})$$

$$D = \begin{bmatrix} 4 & 2 \end{bmatrix} \quad (1 \times 2 \text{ matris})$$

Şekil 2.

Matrisler genelde büyük harflerle isimlendirilir. Bir matrisin m satır ve n sütünü içindeki her bir skaler'e matrisin "elemanları" denir. Matris elemanları genelde matrisi adlandıran büyük harfin küçük hali ve indexler ile isimlendirilir. Matris elemanları hangi satır ve sütunda olduklarına göre indexlenebilir. Örnegin ilk örnegimizdeki A matrisindeki bazı elemanlar:

a12 = 3

a31 = 6

a11 = 2

Bir matrisin ebadını tanımlayan bu "m x n" kavramına matrisin "tipi" diyelim.

Şu ana kadar detaylı şekilde inceleyegeldiğimiz vektörler de matris şeklinde yazılabilir. Örneğin (2, 6, 1) gibi bir 3 boyutlu vektör bir önceki örnekte görülen ortadaki "3 x 1" tipindeki C matrisi olarak yazılabilir.

Matris İşlemleri

Dayanın heyecanlı yerlere geliyoruz. Matrisler üzerinde toplama, çıkarma ve çarpma işlemlerini yapmak mümkündür (bölme demedim dikkat). Bunlar da 3D efektler yaparken hayati önem taşıyan işlemler olduklarından gelin hemen onlara da bakalım.

Matrisleri toplama ve çıkarma

İki matrisin toplanabilmesi veya çıkarılabilmesi için aynı tipte (yani aynı m x n ebadında) olmaları gereklidir. Aynı tipte olmayan matrisler arasında toplama veya çıkarma yapılamaz.

Aynı tipteki iki matrisi toplarken/çıkarırken yapmamız gereken tek şey iki matriste aynı satır ve sütuna denk gelen elemanları toplamak/çıkarmaktır.

$$A = \begin{bmatrix} 3 & 1 \\ 2 & 7 \\ 5 & -3 \end{bmatrix} \quad B = \begin{bmatrix} 2 & 2 \\ 3 & 1 \\ 8 & 2 \end{bmatrix} \quad C = \begin{bmatrix} -4 & 4 \\ 3 & 0 \\ -8 & 1 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 3+2 & 1+2 \\ 2+3 & 7+1 \\ 5+8 & -3+2 \end{bmatrix} = \begin{bmatrix} 5 & 3 \\ 5 & 8 \\ 13 & -1 \end{bmatrix}$$

$$B - C = \begin{bmatrix} 2-(-4) & 2-4 \\ 3-3 & 1-0 \\ 8-8 & 2-1 \end{bmatrix} = \begin{bmatrix} 6 & -2 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Şekil 3.

Matrislerin Çarpılması

Matrislerde çarpma işlemi, toplama ve çıkarmaya göre biraz daha karışık ancak bir kere nasıl yapıldığını öğrendiğinizde

problem olmayacak.

Öncelikle tıpkı toplama ve çıkarma işlemlerinde yaptığımız gibi, iki matrisin çarpılabilmesi için gereken ön koşulu tanıtalım. İki matrisin çarpılabilmesi için birinci matrisin sütun sayısı ile ikinci matrisin satır sayısı aynı olmalıdır. Başka bir deyişle m1 x n1 tipindeki bir matris ile m2 x n2 büyüklüğündeki ikinci bir matrisin çarpılabilmesi için n1 = m2 olması gerekir. Örneğin 3x2 lik bir matris ile 2x6 lık bir matris çarpılabilir. Ancak 2 x 1 lik bir matris ile 4 x 3 lük bir matris çarpılamaz. Aynı tipteki kare matrisler(2x2, 3x3, 4x4 ...) birbiri ile çarpılabilir.

Bu koşulun nedenini anlamak için gelin matris çarpma işleminin basamaklarına bakalım

Elimizde A ve B diye iki tane 2x2'lik matris olsun

$$A=\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} \qquad B=\begin{bmatrix} b11 & b12 \\ b21 & b22 \end{bmatrix}$$

Şekil 4.

Bu iki matrisin çarpımına da C matrisi adı verelim:

A x B = C

Bu iki matrisi çarpmak için yapmamız gereken:

1. Birinci matrisin ilk satırını al
2. İkinci matrisin ilk sütununu al
3. Birinci matristen gelen a1x degerlerini, ikinci matristen gelen bx1 degerleri ile çarp ve sonuçları topla. Başka bir deyişle yukarıdaki örnekte: (a11 x b11) + (a12 x b21)
4. Bu degeri sonuç matrisinde birinci satır birinci sütun sonucu olarak yaz

Bu dört basamağı C'nin bütün elemanları için genelleyeceğiz. Bu genellemeyi yapmak için aynı basamakları şöyle yazalım:

C'nin "m'inci satırında n'inci sütununda olan elemanını" (cmn) hesaplamak için

1. Birinci matrisin m'inci satırını al
2. İkinci matrisin n'inci sütununu al
3. Birinci matristen gelen amx degerlerini, ikinci matristen gelen bxm degerleri ile çarp ve sonuçları topla. Başka bir deyişle yukarıdaki örnekte: (am1 x b1n) + (am2 x b2n)
4. Bu degeri sonuç matrisinde m'inci satır n'inci sütun sonucu olarak yaz

Bu kuralları örneğimize uygularsak:

$$C=\begin{bmatrix} (a11 \times b11) + (a12 \times b21) & (a11 \times b12) + (a12 \times b22) \\ (a21 \times b11) + (a22 \times b21) & (a21 \times b12) + (a22 \times b22) \end{bmatrix}$$

Şekil 5.

İşte birinci matriste bir satırdan alınan değerler ile ikinci matriste bir sütundan alınan değerler bire bir çarpılıp sonuçlar toplandığı için (bir nevi vektörlerdeki "nokta çarpımı" işlemi gibi), bu bölümün başında verdiğimiz ön koşul var. Eğer m1 x n1 ve m2 x n2 tipindeki iki matriste n1 ve m2 eşit değilse bu işlem yapılamıyor.

Şimdi değişik boylarda birkaç matrisin nasıl çarpıldıklarını görmek için aşağıdaki örneklere bakın:

$$\begin{bmatrix} a11 & a12 \\ a21 & a22 \\ a31 & a32 \end{bmatrix} \times \begin{bmatrix} b11 & b12 & b13 \\ b21 & b22 & b23 \end{bmatrix}$$

$$= \begin{bmatrix} (a11 \times b11) + (a12 \times b21) & (a11 \times b12) + (a12 \times b22) & (a11 \times b13) + (a12 \times b23) \\ (a21 \times b11) + (a22 \times b21) & (a21 \times b12) + (a22 \times b22) & (a21 \times b11) + (a22 \times b23) \\ (a31 \times b11) + (a32 \times b21) & (a31 \times b12) + (a32 \times b22) & (a31 \times b11) + (a32 \times b23) \end{bmatrix}$$

$$\begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a21 \end{bmatrix} \times \begin{bmatrix} b11 \\ b21 \\ b31 \end{bmatrix} = \begin{bmatrix} (a11 \times b11) + (a12 \times b21) + (a13 \times b31) \\ (a21 \times b11) + (a22 \times b21) + (a23 \times b31) \end{bmatrix}$$

Şekil 6.

Şimdi birkaç sayısal örnek verelim:

$$\begin{bmatrix} 2 & 4 \\ 1 & 0 \\ 0 & 2 \end{bmatrix} \times \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 2x3 + 4x2 & 2x1 + 4x4 \\ 1x3 + 0x2 & 1x1 + 0x4 \\ 0x3 + 2x2 & 0x1 + 2x4 \end{bmatrix} = \begin{bmatrix} 14 & 10 \\ 3 & 1 \\ 4 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 2 & 2 \\ 4 & 7 & 0 \\ 1 & 4 & 5 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 2 \\ 4 & 7 & 0 \\ 1 & 4 & 5 \end{bmatrix}$$

Şekil 7.

Matrislerle ilgili bazı özellikler:

1. Yukarıda ikinci örnekte görülen köşegen üstündeki elemanların 1 diger bütün elemanların 0 olduğu matrise "Birim Matrisi" denir. Birim matrisi çarpmada "etkisiz eleman"dır. Başka bir deyişle herhangi bir A matrisi Birim matris ile çarpılırsa sonuç gene A matrisi olacaktır.
2. m x n tipinde bir matris ile n x k tipinde bir matris çarpıld-

ığında sonuç m x k tipinde olacaktır.

3. 2x2, 3x3, 4x4 gibi kare matrisler bilgisayar grafik aleminde en çok kullanılan matrislerdir.
4. n boyutlu bir vektör de genelde n x 1 tipinde bir matris olarak yazılabilir.
5. Matris çarpımında "değişme özelliği" yoktur. Yani AxB ile BxA eşit değildir. Hatta ikisinden biri tanımlı bile olmayabilir.
6. Matris çarpımında "birleşme özelliği" vardır. Yani A x (B x C) = (A x B) x C

Vektörel dönüşümlerin matris olarak gösterilmesi

Şimdi matrisleri bir takım sayı öbekleri olmaktan çıkarıp bize faydalı hale getirme zamanı geldi. Matrisler hakkında bu kadar konuşmamızın nedeni bir önceki bölümde bahsettiğimiz vektör dönüşümlerini (yer değiştirme, dönme, ölçekleme gibi) matrisler haline getirebilmemiz. Nasıl mı? Mesela "dönme" dönüşümünü hatırlayalım.

Elimizde P(x1, y1) noktası var. Biz bu noktayı origin etrafında a açısı kadar döndürüp Q(x2, y2) noktasına ulaşmak istiyoruz. Formülümüzü şöyle hesaplamıştık:

x2 = (x1)cos(a) – (y1)sin(a)

y2 = (x1)sin(a) + (y1)cos(a)

Bu formülü matrisler cinsinden de yazabiliriz. Hatırlarsanız vektörlerin matris olarak yazılabileceğini söylemiştik. Dolayısıyla Q ve P vektörlerini matris olarak yazarsak.

$$P = \begin{bmatrix} x1 \\ y1 \end{bmatrix} \qquad Q = \begin{bmatrix} x2 \\ y2 \end{bmatrix}$$

Şekil 8.

şimdi formüle bir daha bakın. Formüldeki iki satırda yapılan işlemleri tek bir matris işlemi ile ifade etmek mümkün mü? Özellikle dikkatimizi çeken noktalar:

1. eşitliklerin solunda x2 ve y2 birlikte Q vektörünü temsil eden matris olarak yazılabilir
2. sağ tarafta da bir takım değerler x1 ve y1 ile çarpılıp sonuç toplanıyor (ya da çıkarılıyor ama o eksi işareti kolayca artıya çevrilebilir)

Öyleyse biz bu formülü aşağıdaki gibi yazabiliriz.

$$\begin{bmatrix} x2 \\ y2 \end{bmatrix} = \begin{bmatrix} \cos(a) & -\sin(a) \\ \sin(a) & \cos(a) \end{bmatrix} \times \begin{bmatrix} x1 \\ y1 \end{bmatrix}$$

$$\text{Dönme}(a) = \begin{bmatrix} \cos(a) & -\sin(a) \\ \sin(a) & \cos(a) \end{bmatrix}$$

Şekil 9.

Q = Dönme(a) x P

İşte burada dönme dönüşümünü bir Matris halinde yazmış olduk. Matrisleri en çok bu amaçla kullanacağız. Yani dönüşümleri ifade etmek için.

Burada şunu sorabilirsiniz. Bir dönüşümü ha matrisle ifade etmişiz , ha önceki formüldeki satırları ayrı ayrı hesaplamışız ne farkeder? Bu çok mantıklı bir soru. Şu an için bir fark görmediniz; farkı açıklayabilmek için birazdan bileşik dönüşümlerden bahsedeceğiz.

Şimdi bir de "yer değiştirme" dönüşümünü hatırlayalım.

Elimizde P(x1, y1) noktası var ve biz bu noktanın D(dx, dy) vektörü kadar yer değiştirmesini istiyoruz. Boylece Q(x2, y2) noktasına ulaşacağız.

x2 = x1 + dx

y2 = y1 + dy

peki bunu nasıl matris olarak yazabiliriz? Tabi bunu

Q = P + D

Diye yazabiliriz elbette. Ancak bir önceki örnekte gördük ki dönme dönüşümünü bir noktaya uygularken dönüşüm matrisi ile noktayı temsil eden matrisi "çarptık". Burada da istediğimiz yer değiştirme formülünü bir çarpım cinsinden ifade etmek.

Malesef bunu mevcut 2 boyutlu vektorler ile yapamıyoruz. Matematikçiler bu sorunu basit bir şekilde çözmüşler. Bakın ne yapmışlar.

$$\begin{bmatrix} x2 \\ y2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x1 \\ y1 \\ 1 \end{bmatrix} = \begin{bmatrix} x1 + dx \\ y1 + dy \\ 1 \end{bmatrix}$$

Şekil 10.

Aslen iki boyutlu olan P ve Q vektörlerinin boyutunu bir artırıp oraya etkisiz eleman olan 1'i yerleştiriyoruz. Böylece 3x1'lik matrislerle artık 2 boyutlu vektörlerimizi temsil ediyoruz. O zaman dönüşümü temsil eden matrisimiz de deminki dönme dönüşümü örneğindeki gibi 2x2 olmak yerine 3x3 oluyor. Dikkat ederseniz

dönüşüm matrisinin 11, 12, 21, ve 22 indexli elemanları birim matris gibiler ve sonuç vektöründeki toplama işlemlerine x1 ve y1 elemanlarının ulaşmasını sağlıyorlar. Matristeki dx ve dy elemanları ise çarpım esnasında P vektörünün 3. Elemanı olan 1 ile çarpılarak sonuc toplamalarındaki yerlerini alıyorlar.

Homojen dönüşümler

Ne yaptığımızı tekrar hatırlayalım

1. vektörleri “nx1” tipinde matrisler olarak gösteriyoruz
2. dönüşümleri “nxn” tipinde matrisler olarak gösteriyoruz.
3. Bir dönüşümü bir vektöre uygulamak için iki matrisi çarpıyoruz.

Q = Dönüşüm x P

Şimdi bu dönüşümden sonuç olarak çıkan Q vektörüne bir başka dönüşüm daha uygulasak

R = Dönüşüm2 x Q = Dönüşüm2 x (Dönüşüm x P)

Birleşme özelliğini kullanırsak:

R = (Dönüşüm2 x Dönüşüm) x P

Yani bunu genellersek şöyle diyebiliriz. Elimizde P noktası varsa ve biz bu noktaya n adet dönüşüm (D0, D1, D2, ... , Dn) uygulayacaksak

R = (Dn x Dn-1 x ... x D2 x D1) x P

İşlemini yapabiliriz.

Yalnız bu işlemi yapabilmemiz için gereken bir koşul var. Uygulanan bütün dönüşüm matrislerinin aynı boyda kare matrisler olması ve vektörün de o boya uygun satır sayısı içermesi. İşte bu yüzden kare matrisler en çok kullanılırlar.

Demin baktığımız iki dönüşüm örneğinde “yer değiştirme” dönüşümü için vektörlere bir fazla eleman eklememiz gerekmişti. Diğer bütün kullanacağımız dönüşümlerde de aynı boyda matris kullanmamız gerektiği için onları da bir büyüteceğiz. Yani 2 boyutlu noktalar için:

1. 3x1’lik vektörler (son elemanı 1 olan)
2. 3x3’lük dönüşümler

3 boyutlu noktalar için ise:

1. 4x1’lik vektörler
2. 4x4’lük dönüşümler

Kullanılır.

Bu şekilde birbiriyle çarpılabilen ve uç uca eklenebilen dönüşümlere “homojen” dönüşümler deniyor.

Şimdi son olarak 3 temel dönüşümün 2 boyutlu vektörlerle kullanılırken aldığı 3x3’lük matris formlarına bir kez daha bakalım.

$$\text{YerDeğiştirme}(dx, dy) = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Dönme}(a) = \begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Ölçekleme}(sx, sy) = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Şekil 11.

Dönüşümlerin uygulanış sırası

Dönüşümlerde sıralama önem taşır. Çünkü aynı dönüşümler farklı sırayla uygulandığında farklı sonuçlar verirler (hatırlayın, matris çarpımında değişme özelliği yoktur demiştik)

Bileşik dönüşümleri genellediğimiz formüle tekrar bakarsak.

R = (Dn x ... x D2 x D1) x P

Bu işlemi iki türlü yorumlayabiliriz:

1. Burada P’ye en yakın olan dönüşüm yani D1 ilk olarak P ye uygulanıyor. Bu dönüşümden sonra çıkan sonuca D2 uygulanıyor vs.
2. İkinci yol ise dönüşümleri “koordinat sistemine uygulanıyor” olarak düşünmektir. Yani normal koordinat sisteminden başlarsak, önce Dn ile yeni bir koordinat sistemine ulaşıyoruz, sonra o sisteme D2 uygulanıyor vs.

Koordinat sistemine dönüşüm uygulamak demek koordinat sistemini tanımlayan eksen birim vektörlerini (mesela iki boyutlu uzay için (1, 0) ve (0, 1)) alıp dönüşümlerle iki yeni eksen elde etmek demektir. Bu iki eksen yeni bir koordinat sistemi tanımlar.

Değişik kişiler bu iki yorum arasından farklı seçimler yapabilir. Siz hangisini seçerseniz seçin, sıralamanın önemli olduğunu unutmayın.

Neden bütün bu matrislerle uğraşıyoruz

Artık bu soruyu cevaplayabiliriz. Hatırlarsanız amacımız pekçok nokta ile modellenmiş bir objeyi almak (bir oyundaki asker veya bir demodaki küpler) ve bir dönüşüm uygulayıp, yeni noktaların konumunu hesaplamaktı. Obje'deki nokta sayısı yüzbinlerce olabilir. Biz n tane dönüşümü yüzbin tane noktaya uygulamak ve sonuçta yüzbin yeni noktayı hesaplamak istiyoruz.

Bunu da hızlı yapmak istiyoruz. Çünkü bunu hızlı yapamazsak frame rate'imiz düşecek.

İşte matrisler ve vektörler bize bunu en hızlı yapmanın yolunu sağlıyor. Biz eger matrisleri kullanmıyor olsaydık, her dönüşümü tek tek bütün noktalara uygulayacak ve her dönüşümden çıkan sonuca bir sonraki dönüşümü uygulayacaktık. Eger bir dönüşümü bir noktaya uygulama için geçen zaman t ise, bir dönüşümü yüzbin noktaya uygulamak yaklaşık 100000 t alacaktı. Bu işlemi 5 dönüşüm için tekrarlayınca 500000 t olacaktı.

Oysa matrisler (ve homojen dönüşümler) sayesinde önce matrisleri birbiri ile (doğru sırada) çarpıp dönüşüm matrisini tek bir matrise indirgeyebiliyoruz. Sonra da bu matrisi 100000 nokta ile çarpınca yaklaşık 100000t zamanda pekçok dönüşümü uygulayabiliyoruz.

C++'da Akıllı İşaretçiler

Bilgem 'Nightlord' Çakır

İşaretçiler(pointer), C++ dünyasına aslında C'den gelen bir kavram. C'ye de assembler'dan gelmişti. C++ programlarında pekçok hata bir kenarından işaretçiler ile ilgili oluyor. İşaretçilerle yapılan aritmetik işlemler, işaretçilerin sıfırlanmasının unutulması vs.

Yine C++ programlarında en sık yapılan hatalardan biri de bellek yönetimi hataları. Sistemden alınan ve iş bittikten sonra geri verilmeyen bellek, veya geri verilmiş belleğe bakmaya devam eden işaretçileri kullanmak vs.

İşte bu iki genel hata grubunun kesişiminde bulunan önemli bir bölümünü çözmek için akıllı işaretçiler (smart pointer) geliştirilmiş.

Pekçok farklı, akıllı işaretçi kütüphanesi geliştirilmiş olsa da, bellek yönetimi için şu an standart kullanmanız gereken akıllı işaretçiler STL ve boost içinde yer alanlar (ki boost içinde yer alanlar önümüzdeki yıllarda STL içine eklenecek).

Akıllı işaretçiler, bildiğimiz normal işaretçiler gibi görünüp kullanılabilen, ama belli koşullarda işaret ettikleri nesne üzerinde delete komutunu çağırıp, sisteme geri verilmesini sağlayan (yani akıllı) sınıflardır (aslında sınıf değil birer şablondurlar(template) ama kafanız karışmasın).

Basitten karmaşığa doğru elimizdeki seçenekler

1. boost::scoped_ptr<>
2. std::auto_ptr<>
3. boost::shared_ptr<> ve boost::weak_ptr<>

Normalde bu üç işaretçi tipini doğru şekilde kullanırsanız, programınızın hiçbir yerinde delete komutunu kullanmanıza gerek kalmamalı. Yani hiçbir şeyi açıkça siz delete etmeden ve hiç bellek sızdırmadan C++ programı yazmak mümkün.

Aklınıza eğer akıllı işaretçiler ile ilgili performans kaygıları geliyorsa, hemen bu kaygılarınızdan sıyrılmak için boost.org'daki akıllı işaretçi performans analizi yazılarını okuyabilirsiniz. Göreceksiniz ki pekçok durumda hiç kaygılanmanıza gerek yok (hatta Herb Sutter der ki, bir programı profiler araçlarından geçirip neresinin performans bakımından zayıf halka olduğunu tespit etmeden, böyle gaipten kaygılar duymamak lazımdır). Kısacası anafikir, gerek STL gere boost içindeki akıllı işaretçiler son derece optimize edilmiş ve son derece iyi test edilmiş kod parçalarıdır. Bunlar hakkında performans kaygısı gütmek için, gerçekten neden bahsettiğini çok iyi biliyor olmak lazım.

scoped_ptr

boost::scoped_ptr en basit akıllı işaretçi türü. Yaptığı iş sadece bir tane nesneye işaret etmek ve o nesnenin tek sahibi olup, silinirken de nesneyi de kendisi ile beraber silmek.

Önce probleme bakalım.

```
void hede(){
    T* t = new T();
    // birseyler yap
    delete t;
}
```

Bu koddaki problem şu: Eğer fonsiyon ortadaki işlemler esnasında dönerse delete t komutu çalışmayacak ve t nesnesini sızdırmış olacağız. Bahsettiğimiz geri dönüş birkaç şekilde olabilir:

1. İlk etapta fonsiyon yazılırken orta bölümde koşullu dallardan birinde return komutu ile çıkış yapılmış, fakat oraya delete konmamış olabilir.
2. Başta olmasa bile sonrada kod geliştirilmeye devam edilirken ortaya bir return eklenmiş ve delete unutulmuş olabilir.
3. Kodda bir return olmasa bile çağırılan herhangi bir fonksiyondan istisna(exception) tetiklenmiş olabilir. Bu durumda istisna yüzünden, kontrol, istisna yakalayıcı bir kod bloğuna rastlayana kadar bir üst çağıran fonksiyona çıkmaya devam eder.

Bu üç durumda da t nesnesini sızdırmış oluruz. Oysa ki bu üç durumda da yığıt(stack) içinde yaşayan bütün yerel nesneler yıkıcıları(destructor) çağrılarak yok edilirler. İşte bu yüzden biz de normal işaretçiler yerine scoped_ptr nesnelerini yığıtta yaratırız:

```
void hede(){
    boost::scoped_ptr<T> p( new T() );
    // birseyler yap
}   // fonksiyon kapsam#ndan ç#k#l#rken p siliniyor
    //ve bakt### nesneyi de siliyor
```

Bu şekilde en başta p akıllı işaretçisi yeni T nesnesine bakacak şekilde yaratılıyor ve ilkleniyor(initialization). Ardından fonksiyondan nasıl çıkılırsa çıkılsın, yerel yığıt değişkeni olan p silinirken, yıkıcı metodunun içinde delete çağrılıyor. Böylece fonksiyonda ne olursa olsun T nesnesini sızdırmıyoruz.

Yani akıllı işaretçimiz yaratıldığı kapsam(scope) içinde olunduğu sürece sahip olduğu nesneye işaret ederken, kapsam dışına çıkıldığı anda, kendisi yok olurken, işaret ettiği nesneyi de siliyor. Bu yüzden adı Kapsam İşaretçisi(Scoped Pointer).

scoped_ptr'nin bir diğer önemli özelliği de kopyalanamaması ve atanamaması(assignment). Bunun sebebi scoped_ptr'nin sadece bir objeye sahip olması ve bu sahipliği başka bir scoped_ptr nesnesine devredememesinin özel olarak hedeflenmiş olması.

```
boost::scoped_ptr<T> p1(new T() );
boost::scoped_ptr<T> p2(p1);  // olmaz!!! derleme hatasi verir
boost::scoped_ptr<T> p3;
p3 = p1  // olmaz!!! derleme hatasi verir
```

Bu kısıtlamanın konulmasındaki amaç, programcının yarattığı nesnenin ömrü ile ilgili daha net kod yazmasını sağlayarak, gelecekte olası problemleri önlemek. Örneğin yukarıdaki hede fonksiyonuna yıllar sonra başka bir programcı ilaveler yapmak durumunda kalırsa, orada yaratılan T objesinin ömrünün, o kapsam içinde sınırlı olması gerektiğini hemen anlayabilir. Böylece mesela yanlışlıkla, o ömrü daha uzun bir başka işaretçiye, p'nin değerini atamaz. Çünkü bunu denerse hemen derleme hatası alır.

Eğer bu kısıt olmasa ve bu atamayı yapabilseydi hata yapmış olacaktı. Mesela düz işaretçiler kullansa, atama yapılan daha uzun ömürlü işaretçi çürük(dangling) bir işaretçi haline gelecekti. Akıllı bir işaretçi kullansa da bu sefer T nesnesini, gerektiğinden uzun süre canlı tutmuş olacaktı.

scoped_ptr'ın kopyalanamaz ve atanamaz olması, kullanımı da çok daha kolaylaştırmış oluyor. Bundan sonra alışkanlık olarak aynı kapsam içinde dinamik olarak yaratıp sildiğiniz bütün nesneler için her zaman scoped_ptr kullanmanızı tavsiye ederim.

Bir uyarı daha. scoped_ptr' ler kopyalanamaz ve atanamaz oldukları için std::vector, std::list gibi hiçbir STL kabına(container) konulamazlar. Çünkü STL kapları içlerine konulan nesne sınıflarının kopyalanabilir ve atanabilir olmasını gerektirir. Eğer scoped_ptr'leri STL kaplarına koyabilseydik, yine kapsamının dışında o nesneleri yaşatıyor olurduk, ki bu belirttiğim gibi tasarım hedefine ters. STL kapları ile kullanacağımız akıllı işaretçiler boost::shared_ptr ve boost::weak_ptr olacak.

Son olarak scoped_ptr'ler yerel nesneler dışında bir de pimpl(pimpıl diye okunur) yaklaşımında kullanılır. Bu yaklaşımdan şimdi bahsetmeyeceğim.

Auto_ptr

std::auto_ptr, geçen yazıda bahsettiğimiz boost::scoped_ptr'den bir adım daha yetenekli bir akıllı işaretçidir. Tıpkı scoped_ptr gibi auto_ptr de bir nesnenin sahibidir, ve kapsam dışına çıkılırken o nesnenin silinmesini sağlar.

```
void hede(){
    std::auto_ptr<T> p( new T() );
    p->hodo();
    …
}    // kapsamdan cikilirken T nesnesi silinir
```

Yine bu fonksiyonda ne olursa olsun (erken bir return ya da istisna) auto_ptr sayesinde temel istisna güvenliği (yani istisnalar tetiklendiği takdirde bellek sızdırmama garantisi) sağlanmış olur.

Ancak scoped_ptr'den farklı olarak auto_ptr, kopyalanabilir ve atanabilir. kopyalama/atama durumlarında nesnenin sahipliği el değiştirir. Başka bir deyişle bir nesneye herhangi bir anda yalnız bir auto_ptr sahip olabilir. Bir auto_ptr bu şekilde sahiplik kaybettiğinde resetlenmiş olur.

```
void hede(){
    std::auto_ptr<T> p1( new T() );
    p1->hodo();
    std::auto_ptr<T> p2( p1 ); // su anda artik p1 resetlendi
    p2->hodo();
    std::auto_ptr<T> p3;
    p3 = p2;   // artik p2 de resetlendi
    // sadece p3 nesnemizin sahibi
    p1->hodo()    // !!!!calisma zamani hatasi
    p2->hodo()    // !!!!calisma zamani hatasi
    p3->hodo()    // OK. cunku p3 nesneye bakiyor
}    // p1 ve p2 resetlendigi icin hicbirsey yapmiyor, p3 nesne
```

Dolayısıyla auto_ptr'ler sahip oldukları nesneyi kopyalama/atama yoluyla birbirlerine devredebilirler. Bu sebeple bazı durumlarda fonksiyonlara arguman olarak geçirilmeye, veya fonksiyonlardan geri döndürülmeye çok müsaittirler.

```
std::auto_ptr<T> Uretici(){
    return std::auto_ptr<T>( new T() );
}
void Tuketici( std::auto_ptr<T> p ){
    p->hodo();
}
void hede(){
    Tuketici( Uretici() );
}
```

Üretici ve Tüketici metodlar karşımıza çok sık çıkarlar. Örneğin birbirine mesaj gönderen veya iş atayan sınıflar mesaj nesneleri üretip tüketiyor olabilirler. Normalde bu yapıda nesnelerin üretilip, el değiştirip, tüketilip, silinmeleri pek çok potansiyel bug/sızıntı noktası taşır. Eğer düz işaretçiler kullanılırsa, nesnelerin sahibinin kim olduğu, hangi işaretçilerin ne zaman sıfırlanacağı gibi konular problem yaratabileceği gibi, herhangi bir basamakta olacak bir istisna tetiklemesi de sızıntıya sebep olur. auto_ptr kullanmak bu sorunların hepsini çok zarif bir şekilde çözer. Buna benzer bir problemle bir sonraki karşılaşmanızsa auto_ptr kullanmak aklınızda olsun.

Son olarak, gerek scoped_ptr, gerekse auto_ptr, sınıflarda üye değişken olarak da kullanılabilir. Böyle kullanıldıklarında sınıfın yıkıcı metodu içinde delete çağrılmasına gerek kalmamış olur.

```
class T{
public:
    T() : u( new U() ) {} // u ilkleme listesinde sistemden ali
    ~T(){} // burada delete yapmaya gerek yok cunku nesnenin ka
private:
    T( const T&);  // kopyalanamaz
    T& operator=( const T& );   // atanamaz
    std::auto_ptr<U> u;
};
```

Burada dikkat etmeniz gereken şey, kopya yapıcı ve atama operatörünün private yapılarak, sınıfın kopyalanamaz ve atanamaz hale getirilmesi. Nitekim sınıfın kopyalanmasında izin verseydik, bu sınıftan bir nesneyi başka bir nesneye kopyaladığımız anda, nesnelerden biri sahip olduğu nesneyi kaybedecekti. Çünkü bir nesnenin u auto_ptr'si diğer nesneninkine kopyalanacak, ve sadece bir auto_ptr nesneye bakmaya devam edecekti.

Aynı sebepten ötürü auto_ptr de tıpkı scoped ptr gibi STL kaplarında kullanılmaya uygun değildir. Kopyalanabilir olduğu halde, auto_ptr kopyaları birbirine denk olmadıkları için, STL kaplarının

metodları kendi içlerinde kopyalamalar yaparken auto_ptr'leri istemeden resetlemiş olurlar. Mesela içi auto_ptr'lerle dolu bir std::vector kabına std::sort işlemini uyguladığınızı düşünün.

Bu noktalara dikkat edilerek kullanıldıklarında auto_ptr'ler özellikle üretici/tüketici ilişkileri olan sınıflar için çok zarif ve güvenli kod yazılmasını sağlarlar.

Shared_ptr

Akıllı işaretçiler arasında en güçlü özelliklere sahip olan boost::shared_ptr<>'den bahsetmeye geldi sıra. Bu işaretçi sayesinde en zorlu bellek yönetim ve nesne ömrü problemlerini çözebiliyoruz.

shared_ptr, daha önce bahsettiğimiz scoped_ptr ve auto_ptr'ye temelde benziyor. Yine amacı bir nesneye işaret etmek ve doğru zamanda nesneyi silmek. Ancak diğer iki akıllı işaretçide olan bir nesneye yalnız bir işaretçinin "sahip" olması zorunluluğu yok. Aksine shared_ptr'leri kullanarak bir nesneyi birden fazla işaretçinin sahipliğine veya paylaşımına verebiliyoruz. Başka bir deyişle bir grup shared_ptr aynı nesneyi paylaşıyor ve paylaşan bütün işaretçiler ortadan kalktığında, nesne de siliniyor. Bu paylaşım kopyalama/atama yoluyla oluyor.

```
boost::shared_ptr<T> Yonetici::YeniTYarat(){
    boost::shared_ptr<T> p1( new T() );
    this->mKayitListesi.push_back( p1 );
    return p1;
}
void App::hede(){
    boost::shared_ptr<T> p = mYonetici->YeniTYarat();
    ...
    // p ile birseyler yap
    ...
}
```

Mesela bu örnekte önce YeniTYarat metodunun içinde p1 işaretçisi objeye bakan ilk işaretçi oluyor. Ardından p1'in bir kopyası bir vector kabına atılıyor. Yani artık iki sahip var. Biri p1, digeri de vectör'ün içindeki işaretçi. Sonra metoddan dönülürken, p1 geçici bir değişkene atanıyor (3. sahip ortaya çıkıyor) ve hemen ardından p1'in ömrü doluyor (kapsam sonuna gelindiği için). Bunun ardından hede metodunun kapsamı içinde isimsiz geçici işaretçi, p işaretçisine atanıyor (bu noktada geçici isimsiz işaretçinin ömrünün dolduğunu varsayabiliriz). Artık bu anda yine iki sahip var. Biri hede kapsamındaki p, diğeri mYönetici nesnesinin içindeki vektörde bulunan işaretçi. Hede metodundan çıkılırken de p yok oluyor ve tek sahip kalıyor. Eğer programın ilerleyen aşamalarında mYönetici nesnesi herhangi bir sebeple silinirse (ve en başta yarattığımız T nesnesine bakan başka shared_ptr yaratılmadıysa), o esnada vektörün içindeki shared_ptr'nin yıkıcı metodu T nesnesini silecek.

Görüldüğü gibi, bu şekilde shared_ptr'ler kullanarak nesnelerin sahipliğini istediğiniz kadar çok sayıda yere bölüştürebilir ve hiçbir zaman elinizdeki hiçbir shared_ptr'nin sallantıya düşmeyeceğinden (silinmiş bir nesneye bakmayacağından) emin olabilirsiniz.

shared_ptr'ler yukarıdaki örnekte görebileceğiniz gibi STL kapları ile kullanılmaya uygundurlar. Hatta karşılaştırma operatörleri de tanımlı olduğu için, ilişkisel kaplarda (associative container) yani std::map ve std::set gibi kaplarda da kullanılabilirler.

shared_ptr kullanırken dikkat etmeniz gereken iki kritik nokta var

Birincisi birden fazla argüman alan metodlara shared_ptr geçirirken, her zaman isimli bir shared_ptr kullanmak. Mesela elimizde soyle iki metod olsun

```
int g();
void f( boost::shared_ptr<T>, int );
```

Daha sonra bunları şöyle kullandığımızı düşünelim.

```
boost::shared_ptr<T> p( new T() );
f( p, g() );
```

Bu kullanım yukarıda bahsettiğimiz kurala uyuyor. Aynı kodu aşağıdaki gibi yazmamız ise sakıncalı:

```
f( boost::shared_ptr<T>(new T() ), g() );
```

Buradaki problem şu. C++ standardı bu şekilde yazılan bir kodda new T(), g() ve shared_ptr yapıcısının hangi sırada çağırılacağını tanımlamamıştır. Yani bu kod çalışırken, bu bahsettiğimiz üç parça herhangi bir sırada çalışabilir. Dolayısıyla eger önce new çalışır, ardından g() çalışır ve en son shared_ptr yapıcısı çalışırsa sızıntı tehlikesi vardır. Bu senaryoda eğer g() istisna tetiklerse henüz shared_ptr yaratılmadığı ve kapsamdan çıkılırken yıkıcısı çağırılmayacağı için, new ile alınan T nesnesi sızdırılmış olur.

Bu uyarıyı tam anlamamış olabilirsiniz. İstisna güvenliği konusu derin bir konu ve o konuda daha sonra yazacağım. Ancak siz anlamadıysanız bile, birden fazla argüman alan metodlara shared_ptr geçirirken bu şekilde satır tasarrufu yapmaya çalışmayın.

İkinci uyarı noktası ise nesneler arası döngüsel shared_ptr kullanım problemi. Mesela T cinsinden iki nesnemiz olsun. t1 ve t2 diyelim. T sınıfının üye değişkenlerinden biri de shared_ptr<T> tipine sahip olsun.

```
class T{
public:
    boost::shared_ptr<T> p;
};
void hede(){
    boost::shared_ptr<T> t1( new T() );
    boost::shared_ptr<T> t2( new T() );
    ....
    t1->p = t2;
    t2->p = t1;
}
```

Burada bir şekilde kodda t1 ve t2 nesneleri birbirlerine bakan bir shared_ptr'ye sahip olmuş durumda. Yani bir işaretçi döngüsü oluşmuş. Bu durumda bir problem oluyor. Dikkat ederseniz birbirine bakan böyle iki nesne olduğu zaman, dışarıdan bu nesnelere bakan bütün shared_ptr'ler silinse bile, bu iki nesnenin içlerindeki shared_ptr'ler kaldığı sürece nesneler silinmiyor. Başka

bir deyişle iki nesne birbirini hayatta tutuyor. Örneğin yukarıdaki kodda hede metodundan çıkıldığında t1 ve t2 işaretçileri yok olacak ve bizim elimizde bu iki nesneye bakan bir işaretçi kalmayacak. Bu da bir nevi sızıntı.

İşte shared_ptr'ler böyle döngüsel kullanıma uygun olmadıkları için bu durumlarda döngüleri kırmak için weak_ptr adı verilen boost işaretçilerini kullanıyoruz.

Weak_ptr

Weak pointer ile bir veya birkac shared_ptr'nin baktığı bir nesneye bakan bir işaretçi elde edebiliyoruz. Fakat bu işaretçi shared_ptr'lerin referans sayacını artırmıyor. Örneğin

```
void hede(){
  boost::shared_ptr<int> p(new int(5));
  boost::weak_ptr<int> q(p);
   …
}
```

Burada q adlı weak_ptr yaratıldığı anda hala p işaretçisi nesnenin tek sahibi. Eğer p'nin kapsamından çıkarsak, p yok olduğu anda nesne de silinecek.

Peki weak_ptr'nin normal bir işaretçiden ne farkı var o zaman? Farkı şu, weak_ptr, normal işaretçiler veya shared_ptr gibi -> operatörünü desteklemiyor. Yani şöyle bir kod yazamıyoruz.

```
q->hodo();
```

q'nun gösterdiği nesneyi kullanmanın tek yolu var o da q'dan geçici bir shared_ptr yaratmak.

```
boost::shared_ptr<int> r = q.lock();
r->hodo();
```

eğer bu anda p ölmüş ve nesne yok olmuş ise lock() metodu null döndürecek. Dolayısıyla weak_ptr böylelikle aynı nesneye bakan işaretçilerden biri delete ile nesneyi sildiğinde diger işaretçilerin anlamsız bir adrese bakmaları problemini ("dangling pointer" diye de bilinir) çözmüş oluyor. Aynı zamanda yukarıda bahsettiğimiz, shared_ptr'ler ile oluşan döngüsel sızıntı problemini de çözüyor.

Sonuç

Akıllı işaretçiler doğru kullanmayı öğrendiğinizde çok işinize yarayabilecek araçlardır. Bol bol kurcalamanız tavsiye olunur.

Alternatif Gerçeklik Yaratmak

Bilgem 'Nightlord' Çakır

Oyun dünyası yaratmak demek, çoğu zaman alternatif bir gerçeklik yaratmak demektir. Bu alternatif gerçeklikteki tarihsel, coğrafik, ve politik örgüde ne kadar çok detay ve tutarlılık varsa, o gerçeklik o kadar inandırıcı olur. Aklınızda bulundurmanız gereken en önemli nokta, bir alternatif gerçeklik yarattığınızda bunun, bir oyundan çok daha fazla üründe kullanılabileceğidir. Aynı alternatif gerçeklikte geçen beş farklı oyun, on kısa hikaye, dört çizgi roman ve iki film yapabilirsiniz mesela.

Dolayısıyla yarattığınız alternatif gerçekçiliğe efor yatırmaktan çekinmeyin. Yani burada oyun tasarımı bölümünde söylediğimin bir bakıma tersini söylüyorum. Oyun tasarımında özellikle kişisel projelerinizde oyuna mümkün olduğu kadar az oyun öğesi eklemenizi söylemiştim. Oyun dünyasını yaratırken ise ekleyebildiğiniz kadar detay eklemenizi söylüyorum.

Çünkü alternatif gerçeklik tasarımı, bana göre bu dünyadaki en zevkli işlerden biridir. Eğer oyun yaparken eğlenmek istiyorsanız, bu tasarım eylemi, işinizin en zevkli parçalarından biridir. Eğer yapacağınız ilk oyunda, tasarladığınız bu alternatif gerçeklikteki detayların çoğunu kullanmıyorsanız bile, belki bir sonraki oyununuzun senaryosu o kullanmadığınız detaylara ihtiyaç duyacak ileride.

Peki bu işi nasıl yapıyoruz. Yani alternatif bir gerçekliği tasarlamak hangi steplerle yapılır. Bunu inceleyelim. Burada ben bu konuya dair herhangi bir metod öğretici kitap bulamadıysam da çeşitli yazın ürünlerinin hazırlanmasına dair çalıştığım kaynaklardan derlediklerim ve sevdiğim yazarlarda gözlediğim bazı seçim ve yaklaşımları ele alacağım. Ayrıca yine İKV oyununda oyun dünyasının yaratılış aşamasında gözlemlediklerim (İKV oyun dünyasının tasarımı büyük ölçüde Kürşad Karamahmutoğlu ve Özgür Soner'e aittir) ve bazı çözmemiz gereken problemler de benim fikirlerimi etkiledi.

Ayrıca dikkat çekmek istediğim bir nokta daha var. Her ne kadar ben aşağıda bir süreçten, yani adım adım ilerleyen bir metoddan bahsediyor olsam da aslında alternatif gerçeklik yaratmak böyle adım adım olmak zorunda değil. Ya da bu adımları benim verdiğim sırada takip etmek zorunda değilsiniz.

Nereden başlayacağız. Bana göre bir alternatif gerçeklikte en temelde iki olgu yatar: Fizik kuralları ve Coğrafya. Olabildiğince erken bir noktada bu iki konu hakkında bazı kararları almanız gerekir.

Fizik Kuralları ve Büyü

Yarattığınız alternatif gerçeklikte, bizim dünyamızdaki fizik kurallarının çoğunun olmaya devam etmesi beklenebilir. Yani yerçekimi, ısı ve enerji vs bildiğimiz gibi mi çalışıyor. Yoksa bu temel kurallarda bir değişiklik yapıyor musunuz. Buralarda yapacağınız olası değişiklikler bazı çok yaratıcı yeni oyun fikirlerini ortaya çıkarabilir. Ancak bunu oyunu kapalı ve mantıklı tutarak kotarabilmek kolay değildir. Özellikle, yerçekimi, enerji ve zamanın akışı gibi konularda bildiğimiz kuralların dışına çıkacaksanız bunun oyununuzu oynayan (veya hikayenizi/çizgi romanınızı/filminizi izleyen) kişilerin algılayabileceği ve öğrenebileceği bir şekilde kotarılması gerekir. Bir alternatif gerçeklikte izleyiciye, bilmediği bir farklı fizik kuralını öğretirken, basit ve dandik durumlara düşmek de çok kolaydır. Bundan başka bir yazıda bahsederim. Komik bir konudur

Şekil 1.

Bu arada böyle temel fizik kuralı değişikliğinin çok başarılı bir örneği Braid adlı indie XBox 360 oyununda var. Bir göz atın.

Fizik kuralları derken aslında daha çok büyü mekanizmalarından bahsediyorum. Yarattığınız alternatif gerçeklikte büyü var mı. Büyü, fantastik ortaçağımtrak oyunlardaki tanıdığımız ateş büyüsü, yıldırım büyüsü gibi şeyler olabileceği gibi, daha uzay çağı kılıklı ortamlardaki jedi güçleri gibi bir formatta da olabilir. Yani büyü derken genel olarak, dünyadaki her adamın yapamadığı ve bizim dünyamızdaki bildiğimiz fizik kurallarının ötesinde olan bilimum yeteneklerden bahsediyoruz.

Çoğu oyunda büyü olayının yapısı ve sınırları da belirli kurallar dahilinde tanımlıdır. Yani belli sayıda ve tipte büyü yapılabilir. Bu büyüler kudret(mana) gibi bir kaynak harcar. Yani yapılan her büyünün bir bedeli vardır ve bu sayede bir büyücü sonsuz cephaneli bir makineli tüfek gibi etrafına ateş topları ve yıldırımlar yağdıramaz. Ayrıca her büyücü her büyüyü yapamaz. Çoğu zaman büyüleri yapabilmeniz için belirli seviyelere gelmiş olmanız gerekir. Veya her seviye atlayışınızda büyülerinize bazı puanlar yatırırsınız. Böylece iki büyücü aynı büyüyü yaptığında etkisi farklı olabilir vs. Eğer hayatınızda herhangi bir frp veya benzeri oyun oynamışsanız zaten bu kavramlara yabancı değil-

sinizdir.

Dikkat edeceğiniz nokta bütün bu büyü mekanizmalarının dengelenebilir ve tanımlı bir şekilde ortaya dökülmüş olmasıdır. Karakterlerin özel güçlerinin bir sınırı olmalıdır.

Hemen bunu çok kötü bir örnekle destekleyelim. Örneğin Süpermen bana göre kabus gibi bir karakterdir. Gücünün sınırı yoktur. Bir arabayı kaldırırken de aynı derecede zorlanır, bir dağı kaldırırken de. Kısacası ne idüğü belirsiz, tanımsız ve kazma bir yazınsal örnektir. Bu tarz tanımsızlıklardan kaçının.

Büyü konusunun alternatif gerçeklikte nasıl bir tanımsal bütünlüğü olması gerektiğine dair bir de iyi örnek vereyim. Ursula K. LeGuin'in Yerdeniz serisinde büyü olayı benim çok hoşuma giden bir şekilde tanımlıdır. Yerdeniz'de var olan her varlığın, yaratılış dilinde bir adı vardır. Bu ad o varlığın ta kendisidir. Bir varlığın gerçek adıdır. Eğer bir varlığın gerçek adını bilirseniz, onun üzerinde kudretiniz olur. Onun hayatı da elinizdedir. Yerdeniz'de insanlar gerçek adlarını ergenliğe girerken bir büyücünün dilinden alırlar ve bu isim kimse tarafından bilinmez. Herkes takma ad kullanır. Ancak çok çok güvendiğiniz birisine gerçek adınızı söylersiniz.

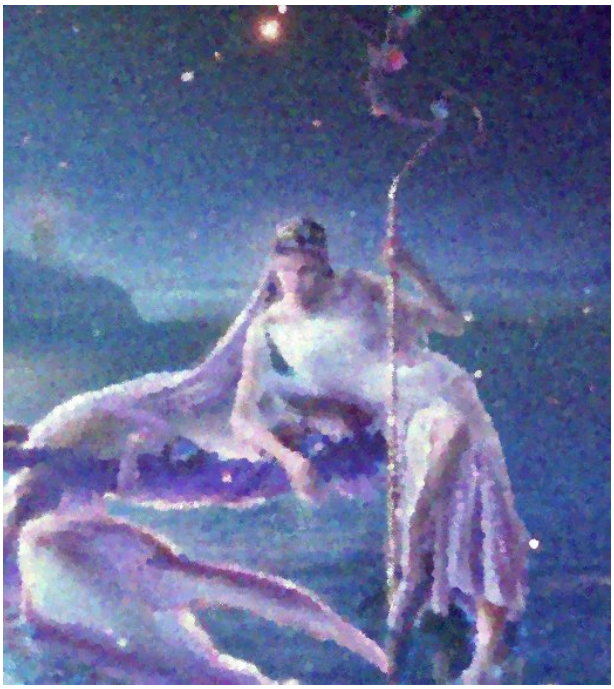
Dolayısıyla, Yerdeniz'de büyü bilimi, bir varlığın gerçek adını anlama, öğrenme bilimidir. Büyücü karakterler bir varlıkla ilgili büyü yapabilmek için onun gerçek adını çözmeye, öğrenmeye çalışırlar.

Bu örnekte, daha baştan edebiyat severleri kalbinden vuran bir kavram var. Kelimelerin gücü. Bir nevi kalemin kılıçtan keskinliği. Ancak aynı zamanda bu yapı büyü kavramının sınırlarını ve karakterlerin yaptıkları şeylerdeki amaçları da tanımlıyor. İşte bu bir alternatif gerçeklikte benim bugüne kadar gördüğüm en çarpıcı büyü tanımıdır.

Genelde bu büyü kararını çok başlarda alacaksınız. Bu yarattığınız alternatif gerçeklikteki pek çok şeyi etkileyecek.

Eğer büyü kavramını gerçekliğin parçası yaparsanız ve olaylar bu dünyada geçiyorsa dinlerle ilgili de nasıl bir duruş alacağınızı belirlemek lazım. Yani büyü gücü dinden ve dualardan falan mı geliyor yoksa doğadaki güçlerden mi? Kimi hikayelerde büyücüler hristiyan öğeler kullanır (ki ben buna kıl olurum). Kimilerinde ise doğadaki güçler büyünün kaynağıdır (ki bu daha pagan bir yaklaşımdır). Genelde büyünün kaynağını doğal güçlere bağlarsanız, çok farklı çeşitlerde lokasyona ve elementlere bağlı binbir türlü büyü uydurabilirsiniz. Bu yüzden genelde bu yaklaşım tercih edilir.

Genel olarak oyun dünyasına üç semavi dinden öğeler katmak kötü bir pazarlama fikridir. Nitekim din hassas bir konudur ve genel olarak oyun yapımında bulaşmamayı tavsiye ederim. Çözüm olarak dinden ya hiç bahsetmeyebilirsiniz, ya da zaman periyodu olarak yeterince geriye giderseniz üç dinin zaman aralığını atlatabilirsiniz. Zaten her üç semavi dinde de geçmişte büyünün olduğundan bahsedilir (Hz. Musa'nın firavunun büyücüleri ile yarışması vs.). Yani mesela M.Ö 2000'lerde geçen ve büyü içeren bir hikaye yapsanız, dinlerle de çelişmemiş olursunuz. Bir diğer sık kullanılan teknik de, hikayenizi son buzul çağından da geriye atmaktır (mesela Conan). Sonuç olarak temelde bu işin eğlence amaçlı olduğunu ve öyle değerlendirilmesi gerektiğini unutmayın.



Şekil 2.

Coğrafya

Coğrafya alternatif bir gerçeklikteki en önemli etkenlerden biridir. Alternatif dünyanız nasıl bir gezegen veya gezegenler coğrafyasında geçiyor. Örneğin Tolkien'in Orta Dünyası gibi tek bir kıta mı. Yoksa Yerdeniz gibi bir adalar grubu mu. Asimov'un galaktik imparatorluğu gibi milyonlarca gezegene dağılmış ama tek akıllı türün insan olduğu bir evren mi. Yoksa StarWars'daki gibi binlerce gezegen ve binlerce farklı tür mü.

Şekil 3.

Özellikle fantazi romanlarında ve oyunlarda her zaman ilk sayfada bir harita olması dikkatinizi çekmiştir mutlaka. Hiç bunun sebebini düşünmüş müydünüz. İşte sebep coğrafyanın bu denli önemli olması. Coğrafya, bütün dünyada yaşayan toplumların, kaynaklarını, hedeflerini, ilişkilerini tanımlıyor.

Mesela Tolkien'in Orta Dünyasında tek bir kıta olması, oradaki bütün toplumların birlikte gelişmesini sağlıyor. Sürekli iletişim halindeler. Bazen savaş bazen ticaret formunda ama sürekli bir etkileşim var. Bu yüzden mesela teknolojik olarak aşağı yukarı aynı seviyedeler. Sadece mesela hobbitler, dünyanın çok da önemli olmayan bir köşesinde yaşadıkları için biraz kopuklar. Keza denizcilik, gemiler, deniz ticareti veya deniz savaşları nispeten düşük öneme sahip.

Öte yandan Yerdeniz adalardan oluşan bir dünya olduğu için orada denizcilik çok önemli. Hikayelerin önemli bölümleri denizde geçiyor. Deniz ticareti, ve ticaret gemileri önemli olaylar ve karakterleri barındırıyor. Mesela büyücüler genelde gemileri hava şartlarına karşı koruyor ve bu onlara önemli iş imkanları sağlıyor. Aynı zamanda o dünyada toplumlar birbirinden daha kopuk. Kimi adalar var ayda yılda bir dışarıdan bir ziyaretçi geliyor falan.

Coğrafya aynı zamanda toplumlar arası çıkar savaşlarını da etkiliyor. Örneğin Orta dünyada cüceler ve elfler arasındaki problemler, cücelerin dağlardaki madenciliği ile ilişkili olabiliyor. Keza doğulu insanların Sauron'un hizmetine girmesindeki sebeplerden biri bu toplumların Gondor'un sahip olduğu zengin doğal kaynakları istemeleri. Keza Saruman'ın yaptığı endüstrileşme, yanındaki ormanları yok ettiği için Entleri kızdırıp savaşa sokuyor. Yani coğrafya doğal kaynakların konumlarını tanımladığı için de toplumlar arası ilişkileri şekillendiriyor.

Yarattığınız alternatif dünyanın coğrafyasını oturtmak bir oturuşta yapılabilecek bir iş değildir çoğu zaman. İteratif olarak yapmak genelde daha iyi sonuç verir.

Gelecek yazıda yarattığınız fizik kuralları ve coğrafya içinde toplumlar oluşturmak ve onların tarihlerini şekillendirmekten bahsedeceğim.

Alternatif Dünyalarda Toplumlar ve Tarihleri

Bilgem 'Nightlord' Çakır

Bir oyunun, ya da fantastik hikayenin geçtiği alternatif bir dünya yaratırken, fizik kuralları ve coğrafyaya karar verdikten sonra artık o dünyada yaşayan halkları ve onların tarihlerini oluşturabilecek noktaya gelirsiniz. Bu halkların bazı karakteristik özellikleri, kültürleri, eğilimleri, inanışları ve politik yapıları, o halkların tarihçeleri ile çift yönlü bir sebep sonuç ilişkisi içindedir. Bu alternatif gerçekliği yaratan kişi olarak, dünyanızda yaşayan halkların kendi içlerinde nasıl geliştiklerini, ve birbirleriyle olan ilişkilerini, özellikle çıkar çatışmalarını ve dayanışmaları iyi işleyebilirseniz dünyanıza büyük bir derinlik katarsınız.

Başlıca alacağınız karar, halkların türleridir. Türden kastım biyolojik tür. Yani dünyanızdaki bütün halklar insan mı. Yoksa Tolkien sonrası bütün fantazi dünyaların yüzde doksanını kaplayan, elfler, cüceler, orklar ve troller de mi var. Belki de hiç insan olmayan tamamen yabancı başka türler var hikayenizde.

Burada bir parantez… Lütfen artık 2011 yılında yeni bir dünya oluştururken elfler, orklar, hobbitler kullanmayın. Lütfen lütfen. Blizzard'ı ve World of Warcraft'ı örnek almayın bu konuda. Warcraft neredeyse 15 yıllık bir franchise ve Warcraft 1 çıktığında orklar ve elfler henüz bu kadar suyu çıkacak derecede kullanılmamıştı. O seriyi devam ettirdikleri için Blizzard bugün hala elfli orklu bir alternatif dünyayı kullanabiliyor. Ancak bugün yeni bir alternatif evren yaratıyor olsalar, sanırım onlar da Tolkien halklarını kullanmazlar. Dünya bu halkların doldurduğu alternatif evrenlere fazlasıyla doldu. Onun yerine başka mitolojilere veya tamamen yaratıcılığınıza dayanmayı tercih edin. Hele Anadolu'lu bir insanın kendi topraklarında dayanabileceği binlerce mitolojik yaratık ve halk varken, aynı beş İskandinav mitolojisi tabanlı halkı kullanmaya hiç gerek yok.

Eğer dünyanızdaki halklarda birden fazla tür bulunuyorsa, almanız gereken bir diğer karar bu türlerin hangilerinin melez çocuklar üretebileceği. Aslında biyolojide tür (yani species) denen grup sadece kendi içinde üreyebilir. Ancak çoğu fantazi dünyada bu göz ardı edilir. Örneğin Tolkien'in Orta Dünyası'nda bir elf ve bir insanın çocukları olabilir (Beren ile Luthien veya Aragorn ile Arwen gibi). Bu tarz melezlere izin verirseniz türler arası aşk ve iki türün güçlü yönlerine sahip önemli karakterler elde edebilirsiniz ki bu bir yazar olarak elinize kullanılabilecek bir çuval dolusu malzeme sağlar.

Dünyanızdaki halkları belirledikten sonra, her bir halkın temel fiziksel özelliklerini netleştirmeniz gerekir. Diyelim ki dünyanızda yaşayan Tolaklar diye bir ırk var. Bir Tolak neye benzer. ortalama boyları ve kiloları ne kadardır. Ne kadar güçlüdürler (bir insana nazaran). Nasıl beslenirler. Hangi iklim koşullarında rahat yaşarlar.

Diyelim ki Tolaklar, insana benzeyen ancak gözlerinin arasındaki mesafe çok daha fazla olan (yani gözleri kafalarının iki yanına doğru açılmış) 4 kollu canlılar olsun. Bir insandan biraz daha iriler, daha uzun boylu ve kilolular. Buna orantılı olarak da daha güçlüler. Ağırlıklı olarak balık ve su bitkileri tüketiyorlar yiyecek olarak ve bol miktarda su. Daha çok soğuk bölgelerde yaşamayı tercih ediyorlar çünkü çok sıcakta vücutları çok su kaybedip güçsüz düşüyor.

Şekil 1.

Bu demek oluyor ki Tolakları haritanızın soğuk bölgelerine yoğunlaştırmanız lazım. Aynı şekilde Tolaklar su kaynaklarına diğer toplumlardan daha bağımlılar. Şehirleri her zaman büyük akarsuların yanında oluyor.

Bu noktada bazı sosyal soruları sormaya başlıyoruz. Tolaklar sosyal olarak nasıl bir oluşuma sahip. Erkek ve dişinin toplumdaki yeri eşit mi. Yoksa ana erkil veya ata erkil bir kültürleri mi var. Ne kadar tarım, ne kadar avlanma, ne kadar ticaret yapıyorlar. Toplumda mesleklere göre bir sınıflanma var mı. Dinleri nasıl bir din. Dilleri nasıl bir dil. Yazılı tarihleri var mı.

Bu soruların cevapları birbirleri ve coğrafya ile iliskiler doğuruyor. Örneğin diyelim ki, Tolaklar ana erkil bir toplum. Ağırlıklı olarak tarım ve az miktarda ticaret ile uğraşıyorlar. Dinlerinde su kutsal. Su tanrısı To'ya tapıyorlar. Su onlar için çok önemli ve hayat kaynağı olduğu için inanışlarının böyle şekillenmesi normal. Zaten Tolak kelimesi onların dilinde To'nun kızı anlamına geliyor. Yani kendilerini To'nun kızları olarak adlediyorlar. Bu da ana erkil kültürleri ile örtüşüyor. Dilleri genelde kalın sesli harflerden oluşan kelimelerden oluşuyor. Buna göre dünyadaki Tolak şehirlerinin veya Tolak karakterlerinin isimleri bu dil ile uyumlu. Aktalot, Ralama ve Havgalot mesela bazı Tolak köylerinin isimleri olabilir. Kalor, Boda ve Vano üç Tolak karakteri olabilir. Öte yandan İlezörion diye bir Tolak kasabası olmamalı.

Öyleyse Tolak şehirleri soğuk bölgelerde su kenarlarında genellikle küçük tarım kasabaları şeklinde. Soğuk bölgeler genelde dağlık olduğu için, dağlık bölgelerde dağınık küçük köylerden oluşan bir toplum. Yani büyük metropolisleri yok. Buna bağlı olarak teknolojileri çok ileri değil. Ancak su büyüleri konusunda derin bir irfanları var. Aralarında az sayıda da olsa kuvvetli su büyücüleri çıkıyor.

Dikkat ederseniz, coğrafyadan başlayıp halkların fiziksel özelliklerini ve genel kültürel yapılarını oluşturdukça her yeni adımda bazı şeyler kendiliğinden ortaya çıkmaya başlıyor. Örneğin suya ve soğuk iklime olan biyolojik bağımlılıklarının sonucu olarak dağlık bölgelerde büyük şehirler kuramayacaklarına ve bunun sonucu teknolojik olarak kısıtlı kalacaklarına varıyoruz.

Başka bir deyişle bu yaratma süreci biraz tasarım ise biraz da keşif. Yani bazı zamanlarda kendimizi o dünyada dolaşan bir ziyaretçi yerine koyuyoruz. Bir takım tasarım kararlarından sonra Tolaklar'ın MaviDağ'daki köyüne gidip dolaşsak neler görürdük. Orada hangi zanaatkarlar var. Tolak köyünün meydanında yürüyorsunuz. Etrafınızda neler görüyorsunuz. Havadaki koku ne. Duyduğunuz sesler ne. Bu gözlemci kimliğini benimseyin ve hayal edin. Tolaklar gerçekten varlar ve yüzyıllardır yaşıyorlar. Siz kafanızda onların hakkındaki bütün detayları zaten biliyorsunuz. Bu detayları keşfetmek için Tolak köyünü gezmelisiniz.

Bu hafif metafizik gibi görünen mekanizma aslında genelde Ursula LeGuin'in kendi yazma metodu olarak bahsettiği metodla örtüşüyor. Çok uzun yıllar bu argümanı anlamadım. Bir insan henüz yaratmadığı bir dünyada nasıl bir gezgin olup detayları keşfedebilir.

Şimdilerde bunu daha iyi anladığımı hissediyorum. Burada bir perspektiften bahsediyoruz. Bazı anahatları düşündükten sonra o dünyada kendinizi bir gezgin olarak hayal ettiğinizde, keşfettiğiniz şeyler aslında, kendi bilinç altınıza ve benliğinize yıllarca işlemiş olan biyoloji, sosyoloji, antropoloji, psikoloji, politika, ekonomi gibi konulardaki bilgi birikiminizin ve dünya görüşünüzün bazı tutarlı ve kaynaşık yansımalarından başka birşey değil. Bu gerçeklik ve tutarlılık neredeyse mistik bir şekilde açığa çıksa da arkasında aslında yıllarca süren yaşamışlık ve okumuşluk birikimi var.

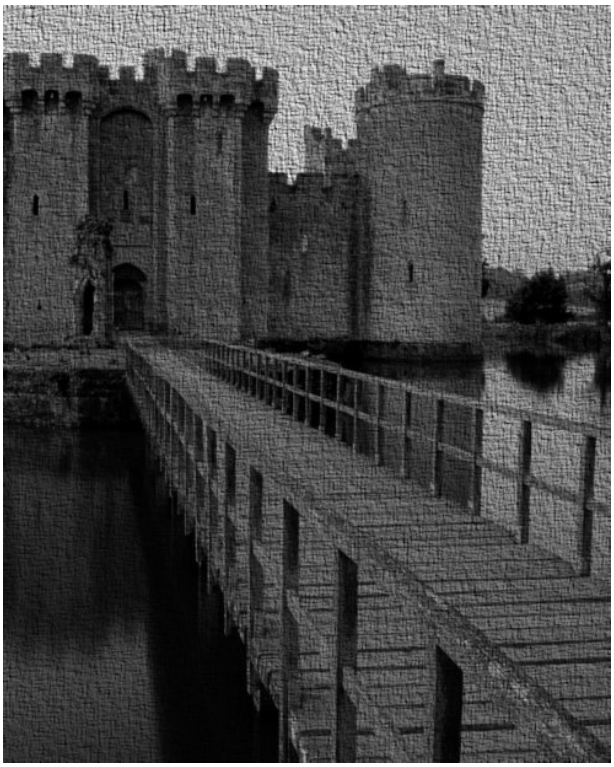
İşte bu yüzden, alternatif evrenler yaratmak isteyen insanların, bol miktarda geniş bir yelpazeden kaynakları okuması ve anlaması gerekiyor. Bu zaman içinde giderek daha iyiye gidecek bir yeti. Ancak bu konuda kendinize yatırım yapmak istiyorsanız, benim tavsiyem, tarih, sosyoloji, dinler tarihi, psikoloji, antropoloji gibi sosyal bilimlerde vakit buldukça birşeyler okumanız.

Gelelim toplumların tarihlerini yaratmaya. Bir toplumun tarihini yaratırken, değişik seviyede detaylandırma hedefleyebilirsiniz. Temelde benim tavsiyem iteratif bir yaklaşım sergilemenizdir. Öncelikle kaç bin yıllık bir tarih (yani yazılı tarih) var. En temel olaylar neler. En temel karakterler kimler

En temel olaylar genellikle coğrafik konum değişiklikleridir. Başka bir deyişle göçler veya işgaller. En temel karakterler de, genelde en temel olaylarda yönlendirici rol oynayan büyük liderler veya toplumun hayatını kökten değiştiren bazı karakterler olabilir.

Mesela Tolaklar, eskiden dünyanın daha güney bolgelerinde yaşarken buzulların erimesiyle Kuzeydeki Soğuk Dağlara göç etmek zorunda kalmış. Bu göçü o dönemdeki Tolak kraliçesi Akta yönetiminde yapmışlar. İlk yerleştikleri kasabaya Akta'nın kasabası, yani Aktalot adı verilmiş. O zamanlar henüz büyü irfanına sahip değillermiş. İlk büyücüleri, yeni yerleşilen Soğuk Dağlardaki su kaynaklarının irfanını ilk çözen Tolak olan Havga imiş. Havga zaman içinde bir Havgalot büyü okulunu kurmuş ve Tolakların su büyülerinde uzmanlaşmasını sağlayacak olan olayların temelini atmış.

Ayrıca bir toplumun tarihçesinde en önemli bileşenlerden biri o toplumun kendi içinde veya diğer toplumlarla yaşadığı bazı çatışmaların da anlatılmasıdır.



Şekil 2.

Mesela Havga'nın büyü yoluyla güçlenmesi Kraliçe Akta'yı ve Aktalot'luları kaygılandırmaya başlamış. Zaman içinde Havgalotlular ve Aktalotlular arasındaki gerginlik tırmanmış ve bir yıl süren bir iç savaş yaşanmış. Savaşı Havgalot kazanmış ancak, Havga, Kraliçe Akta'yı tahtta gözü olmadığına ikna etmiş. Bundan sonra Tolakların yönetimi bir Kraliçe ve onun baş danışmanı olarak bir Su büyücüsü arasında paylaştırılmış. Akta hanedanı bugüne kadar sürmekte ve Havgalotlu büyücüler bugüne kadar baş danışmanlık görevini yapmakta.

Şimdi bu basit tarihçede bir anda topluma yerleşik bir problem eklemiş oluyoruz. Belki Tolaklar şu an barış içinde yaşıyor ama hala iki en büyük Tolak şehri olan Aktalot ve Havgalot arasında bazı gerginlikler ve rekabetler var. Bu gerilim bize yüzlerce değişik alt hikaye sağlayabilir.

Politik gerilim, mutlaka yarattığınız her alternatif gerçeklikte bütün hikayelerinizi besleyebilecek sonsuz bir kaynaktır. Halkların içlerinde veya aralarında Politik gerilim yaratırken, gerilimde tarafların bazı temel ihtiyaçlarının rol oynamasına çalışın. Örneğin Tolaklar Soğuk Dağların doğusunda yaşayan ve madencilik yapan Gizduglu insanlarla bir problem yaşayabilir. Mesela Gizdugluların Maden ocakları Soğuk dağlardaki yer altı su kaynaklarını kirlettiği için Tolaklar ile Gizduglular arasında savaşlar olabilir.

Halkların tarihçelerinde böyle temel politik gerilim öğelerini bir kere iyi kurabilirseniz, oyunlarınıza veya hikayelerinize güçlü bir olay ve karakter kaynağı yaratmış olursunuz. Bu politik gerilimlere bol bol zaman ayırın

Aynı şekilde politik gerilimlerin tersine, halklar arası çıkar ilişkileri de oluşturabilirsiniz. Hatta aynı iki halk arasında bir tarihte savaş olup başka bir tarihte müttefik ilişkiler olabilir. Bu tür karmaşık politik örgüleri kurmak başta zor görünse de bu konularda bol bol beyin fırtınası yapıp deneyler yapmaktan çekinmeyin.

Gördüğünüz gibi bu yazı bünyesinde tamamen sıfırdan Tolaklar halkını uydurduk ve onları coğrafyamıza yerleştirip, pek çok oyun için gerekenden daha detaylı bir sosyal yapı ve tarihçe yarattık. Bu basit örnekte açık bıraktığımız pek çok detayda, daha sonraki iterasyonlarda eklenebilir.

Bu süreci bütün halklar için tamamladığınızda artık elinizde fizik kuralları ve coğrafyası tanımlı bir dünyada yaşayan, ticaret yapan, savaşan, barışan bir grup halk ve bu halkların tarihçeleri var. Başka bir deyişle elinizde bir süredir yaşamakta olan bir dünya var. Bu noktada tekrar gezgin şapkamızı takıp bu dünyada dolaşabilir ve herhangi bir mantıksızlık olup olmadığını gözden geçirebiliriz. Bu noktada yapacağınız en önemli kontrol bu dünyada güçlerin dengeli olup olmadığıdır.

Kısaca dünyanızda yaşayan her bir halka bakın. Bu halk bu dünyada var olabilir mi. Elindeki kaynakları bir diğer halk ele geçirmek ister mi. İsterse bu halk kendini olası bir saldırıya karşı savunabilir mi. Savunamazsa başka bir halkı kendini savunmak için kullanabilir mi. Anlaşmalar yapabilir mi.

Başka bir deyişle bu dünyada bu halkların şu anki hallerinde olmaya devam etmelerine yetecek güç dengesi var mı. Bu denge yoksa bu halklardan biri veya bir kaçı ya yok olmalı ya da yok olmak üzere.

Gelecek yazıda daha mikro kademeye inip, karakterler ve akıştan bahsedeceğim.

Öz Desert Dream - Perde Arkası

Alp 'Domino' Yener

Öz Desert Dream fikrinin ortaya çıkışı bir geyik muhabbeti ile başladı. 7D9'da yaşanan Megablast olayı bizim ilham kaynağımız oldu diyebilirim. "Farklı platformlarda farklı yöntemler kullanılarak nasıl demo yapılır" üzerine dönen sohbetler bir anda yerini "demo parodisi yapsak nasıl olur" fikrine bıraktı. Yapsak mı, yapmasak mı, yaparsak nasıl yaparız düşünceleri gün geçtikçe yerini netleşen fikirlere bırakmaya başladı ve 7DX 2010'a Zomco bir wild entry ile katılacak kararı verildi.

Hiç "hangi demonun parodisini yapsak" diye düşündüğümüzü hatırlamıyorum. Sanırım daha önce bunu bir şekilde aklımdan geçirmiştim ve bu fikir bir şekilde bilinçaltıma yerleşmiş olmalıydı. Kesinlikle Desert Dream'in parodisi yapılacaktı. Başka demolara bakılmadı bile...

Desert Dream parodisi yapmaya karar verdiğimizde Şubat veya Mart ayındaydık. Çekimleri yapmayı düşündüğümüz tüm mekanlar yaz aylarında daha uygun olacağı için, çekimleri yaza kadar ertelemeye karar verdik. Bu sırada bir yandan da projeyi olgunlaştırmak için fikir yürütmeye başladık...

Piramitlerin olduğu bölüm nispeten en kolayı olacaktı. Yatuyu ile birlikte bu kısım üzerinde düşünürken; Çöl hissi verecek bir plajda, kum takviyeli karton piramitler yapmaya karar vermiştik. Aynı şekilde, uzay aracı da kartondan yapılacak ve üzerine gerçek bir karpuz dilimi yerleştirilecekti. Tabii o koca karpuz diliminin karton uzay aracı üzerinde nasıl duracağı konusu henüz netlik kazanmamıştı. "Bir şekilde hallederiz" diyerek, bu sorunun üstünü çekim günü çözüme kavuşturmak üzere kapattık.

Göz açıp kapayıncaya kadar yaz ayları geldi çattı tabi. İş-güç, tatiller, mangal partileri, bu hafta şu planım var, haftaya başlarız bahaneleri birbiri ardına sıralanmaya başladı ve bir de baktık ki yaz ayları bitmiş, güneşli havalar geride kalmış ve partiye sadece bir ay kalmıştı!

Demodaki birçok part için henüz hiçbir şey düşünmemiştik bile. Nasıl olsa piramitli bölümün planı yapılmıştı, gerisi bir şekilde halledilirdi ama kazın ayağı hiç de öyle değildi tabi. Demodaki efektlerin gerçek hayatta 'aslını andıracak' ve 'esprili' bir şekilde uygulanmasının çok da kolay olmadığını anlamamız pek uzun sürmedi. Kara kara düşünmeye başladık. Bu arada bazı acı gerçekleri de fark ettik: Yakınlarımızda çöl hissi verebilecek bir plaj yoktu ve hiçbir yerde karpuz kalmamıştı!

Planlarda değişiklik yapmak zorundaydık. Piramitleri de kartondan oluşturmaya karar verdik. Hem böylece demodaki vektör grafiklere daha yakın bir görüntü elde etmiş olacaktık. Bu konu tamamen netliğe kavuştu. Sıra geldi diğer partlar için birşeyler düşünmeye... Scrolltext içeren partlar için çözüm kolaydı. Teksti kağıtlara yazıp, kameranın önünden kaydıracaktık. Bunu uygulayacağımız iki bölüm vardı. Biri orijinal demonun ilk bölümündeki zoomer içeren tekstti. Diğeri de ikinci disketin hemen başındaydı. İkinci disketteki teksti koca koca puntolarla toplam 5-6 adet A4 kağıdı kaplayacak şekilde yazıcıdan çıkarttım. (Aslında elle yazmalıydım diye düşünüyorum şu an.) Işık kaynağı olarak da en yakın oyuncakçıdan bir laser pointer temin ettik ve laser pointerı yazının etrafında döndürmek suretiyle bu partı hallederiz dedik. Ama bunlar en kolay kısımlardı. İşin ne kadar zor olduğunu diğer partlar için hangi malzemelerle ne yapacağımızı düşünmeye başladığımızda anlamıştık.

Aslında Öz Desert Dream'i orijinal demodaki senaryo sırasına uygun şekilde hazırlamadık. Hazırlanma aşaması karışık olduğu için, bu esnada başımızdan geçenleri sırasıyla yazmak, okuyan için çok da ilgi çekici olmayabilir diye düşündüm ve yazının buradan sonraki bölümünü orijinal demoya paralel olarak yazmaya karar verdim. Böylece, (eğer bu yazıyı buralara kadar okuyan biri olursa) bir yandan Öz Desert Dream'i (ve/veya Desert Dream'i) part part izleyip, bir yandan yazıyı okuyabilir ve her partın hikayesini, hazırladığımız videoyu izlerken okuyup kafasında daha net canlandırabilir.

GÖZLER

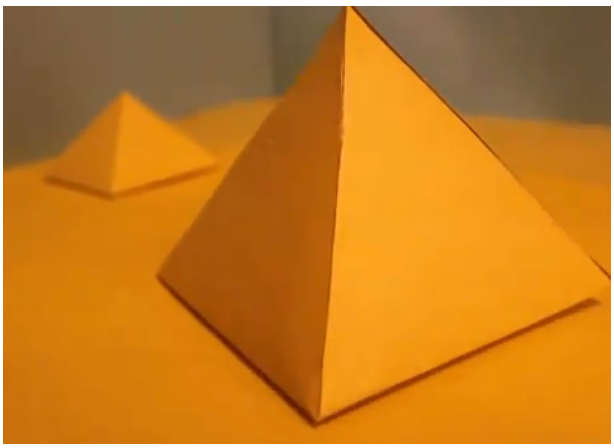
Şekil 1.

Orijinal demo bir çift gözün açılıp kapanması ile başlıyor. Bu en kolay bölümlerden biriydi ve en son hazırladığımız bölümlerden biri oldu. Zira çekimler esnasında kullandığımız kartonlardan birini bu bölüm için delmek durumundaydık ve bu olayın ardından o kartondan bize bir daha hayır gelmeyecekti. Tüm çekimlerimizi tamamladıktan sonra sıra bu bölüme geldi. Çekimler sırasında Öz Desert Dream'i hazırlayan ekip dışında birkaç arkadaşımız daha zaman zaman bizimle birlikte oldular. Kimileri sadece yaptıklarımızı şaşkın bakışlarla izleyip, bizimle dalga geçmeyi tercih etti, kimileri de ucundan kıyısından bu muhteşem esere(!) katkı sağlamak istediler. Başlangıçta düşüncem gözlerin açılıp kapandığı bölümde kendi gözlerimi kullanmak şeklindeydi ama kartona göz deliklerini açtıktan sonra nasıl olduysa bir anda karton, avukat arkadaşım Herol'un eline geçti ve deliklerden çıkarttığı gözleri o kadar çekici, o kadar anlamlı bakıyordu ki, dayanamadık ve onun gözlerini kullanmaya karar verdik. Kaşla göz arasında bu bölümü çektik. Bu vesileyle Zonguldak Barosu'nun değerli avukatlarından Herol arkadaşıma tekrar teşekkür etmek isterim. Bu arada çekimler sırasında bizi izleyip dalga geçenler arasında, görevi nedeniyle resmi protokole dahil olan bir büyü-

ğümüz de vardı ki kim olduğunu ve görevini söylememem daha uygun olur sanırım. Protokollü mrotokollü enteresan bir çekim süreci oldu, o kadarını söyleyeyim... :)

PİRAMİTLER

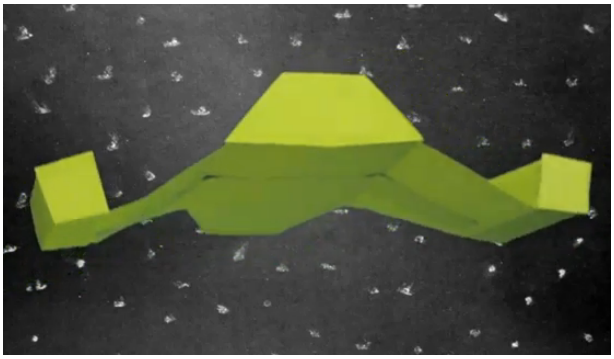
Desert Dream'i hepiniz görmüşsünüzdür ama şimdi biri size sorsa demodaki partları, efektleri tek tek hatırlamayabilirsiniz. Demonun tamamını hatırlamasanız bile Desert Dream dendiğinde izleyen herkesin aklında piramitli bölüm canlanıyordur eminim. Bu bölüm demonun en can alıcı bölümü ve parodi hazırlanırken diğer bazı partlarla ilgili sıkıntılar olsa bile, piramitli bölüm çok iyi hazırlanmalıydı. Aslına uygun olmalı ve tabii ki esprilerle süslenmeliydi. Çekimlerin nasıl yapılacağını kararlaştırdık ve malzemeleri temin etmek için bir kırtasiyeciye girdik. “Karton istiyoruz” dedik. Dükkancı amca bize kartonları gösterdi. Yeterince büyüklerdi ve birkaç değişik renk seçeneğimiz de vardı. Uzay için siyah, çöl ve piramitler için sarı, gökyüzü için mavi, karpuz için kırmızı ve yeşil renkli kartonlardan aldık ve bu sırada bir yandan Yatuyu ile “şu obje için şu kadar karton gerekir”, “bu sahnede bu kadar filanca renk karton kullanırız” diye konuştuğumuz için dükkancı amca merakla “bu kartonları nerede kullanacaksınız” diye sordu. İşte zurnanın zırt dediği yer burasıydı! Adama verebilecek mantıklı ve anlaşılabilir bir cevabımız yoktu. “Demo parodisi yapacağız” desek adama bunun ne olduğunu anlamak için soracağı diğer soruların sonucunda bilgisayar tarihini anlatmak durumunda kalabilirdik. Ben kara kara düşünürken Yatuyu, “bir yarışmaya katılacağız, çekim falan yapacağız” deyince adam daha fazla bilgi talep etmedi ve “yarışmada başarılar, Allah muvaffak etsin” diyerek görevini tamamladı. O anda, yaptığımız işin ne olduğunu çevremizdeki insanlara nasıl açıklayacağımızı bilmediğimizi anlamış olduk ki; “siz ne yapıyorsunuz yahu” sorusu Öz Desert Dream'in hazırlanışı sırasında en çok karşılaştığımız ve kimseye tatmin edici bir cevap veremediğimiz bir soruydu. Kırtasiyeci adam da çocuklarımızın ödevi için birşeyler hazırlayacağımızı düşünmüştür sanırım. Koca koca adamların ne işi olur o kadar karton ve tutkalla!



Şekil 2.

Ekip hazırdı. Ben karton objeleri hazırlayacaktım. Yatuyu çekimleri yapacaktı ve daha sonra Squidward'a yaptıklarımızı montajlaması için baskı yapacaktık. Squidward'ı ikna etmeyi başaramazsak montaj da bana kalacaktı. Plan bu şekildeydi... Malzemeler tamamlandı ve ilk çekim günü geldi çattı. Akşam iş çıkışı Squidward ve avukat arkadaşım Bertem ile birlikte benim evde toplandık. Yatuyu'nun mesaisi daha geç bittiği için o bize sonradan katılacaktı. Yatuyu gelene kadar ben objeleri hazırlarım, belki sonra Squidward bana yardım eder ve bir iki sahneyi çekeriz diye düşünüyordum. Fakat karton ve makası elime aldığımda bu objeleri o kadar da kolay hazırlayamayacağımı anlamam çok uzun sürmedi. Birden ilkokul yıllarıma döndüm ve kesip biçme konusunda ne kadar başarısız bir öğrenci olduğumu hatırladım. Acaba Yatuyu bu işi benden alır ve beni bu eziyetten kurtarır mı diye kara kara düşünmeye başlamıştım ki; Squidward şu cümleyi kurarak hayatımı kurtardı. “Ne yapılacak? Ver makası Bertem'e, o yapsın. Bu adam maket falan yapıyor, ödülleri bile var. Kafayı kesip biçmeyle bozmuş, manyak la bu Bertem...” İşte bu müthiş bir haberdi. Bertem'in böyle bir yeteneği olduğunu bilmiyordum. O ana kadar beni izleyip, “ne yapıyosunuz olm siz... Başka işiniz gücünüz yok mu... Ya bırakın bu işleri de ordan bana bi bira daha getirin...” gibi sözler sarf etmekten başka bir iş yapmayan Bertem, kaşla göz arasında Zomco üyesi olmuş ve projedeki en önemli isim haline gelmişti. :)

Bana ne yapılacağını sordu. Ben de ona “işte şu şu boyutlarda üç piramit yapılacak” dedim. Başka birşey gerekip gerekmediğini sordu. Çekinerek; “aslında bir de uzay gemisi var” dedim. Desert Dream'i açtım ve hazırlanacak objeleri gösterdim. Objelere şöyle bir baktı ve benden cetvel istedi! “N'apıcaksın ya cetveli, alt tarafı üç piramit, bi de kıytırık uzay gemisi” dediğimi çok iyi hatırlıyorum. Adam cetvelsiz yapmam diye tutturdu. (Sanatçı kaprisi) Ne yaptım, ne ettim biryerlerden cetvel bulup getirdim. Ölçtü, biçti, kartonların sağını solunu işaretledi, çizdi falan... Ben bir yandan ne yaptığına bakıyorum tabii... Sağdan bakıyorum, soldan bakıyorum, hiçbirşeye benzemiyor. “Bu böyle olur mu yahu” demek gafletinde bulunduktan sonra sağdan katladı, soldan katladı, ordan kesti, burdan yapıştırdı ve az önce oracıkta hilkat garibesi gibi duran karton parçası bir anda uzay gemisine dönüşüverdi. Bir gemiye bakıyorum, bir ekrana bakıyorum, ancak bu kadar olur! Adam demodaki uzay gemisini bire bir yapmıştı!



Şekil 3.

Bertem bu yaptığına daha sonra pişman olacağının o an için farkında değildi tabi. :)

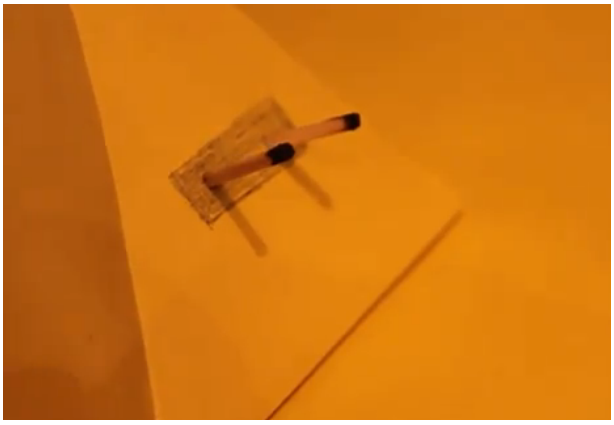
Artık çekimlere başlayabilirdik. Siyah kartonlardan birine sabun marifetiyle yıldızlar çizdim ve işte uzayımız hazırdı! Uzay gemisini iki noktasından misinalara tutturdum ve uzaktan kontrol edilebilir hale getirdim. (Bu arada misina istediğim av malzemeleri dükkanının adını hatırlayamadığım sahibinin “misinayı nerede kullanacaksınız, hangi kalınlıkta vereyim” sorusuna; “bir çekim işi var, kuklalar falan, onlara tutturacağım, en incesinden olsun” cevabını verdiğimde; “filimci arkadaşıma ikramım olsun” diyerek ısrarla benden para almadığı için teşekkür edeyim.)

Uzay dekorumuz hazır, gemimiz dekorun önünde hazır ve motörrrr!!! İlk sahneyi çektik, izledik ve sonuç oldukça tatmin ediciydi. O anda 7DX'te bu video gösterildiğinde alacağı tepki gözümün önünde canlanmıştı...

Çekimlerin bir kısmı benim evimde, bir kısmı da Bertem'in evinde gerçekleştirildi. Benim evimdeki çekimler sırasında bir yandan alkol ve ardından gelen sohbetler, Bertem'in evindeki çekimler sırasında ise Bertem'in aynı apartmanda yaşayan akrabalarının sık sık bizim ne yaptığımıza bakmak için gelip bizimle dalga geçmeleri, çaylar çörekler ikram etmeleri çekimlerin aksamasına ve zamanımızın daralmasına neden oluyordu. Kaç kişinin “siz ne yapıyorsunuz” sorusuna cevap vermek için debelenip, tatmin edici bir cevap veremediğimizi hatırlamıyorum bile. :)

Geldik piramitlerin olduğu bölüme: Piramitleri sırasıyla zemine yerleştirdik. Zeminde sarı çöl kartonu ve kadraja girecek her alana mavi gökyüzü kartonunu yerleştirdikten sonra çekimleri gerçekleştirdik. Fena değillerdi. Ama enteresandır, hiçbirimiz yanlış lens kullandığımız için renklerin düzgün çıkmadığını fark edememiştik. Videoya bakarsanız piramit sahnesinde gökyüzünün siyah olduğunu görürsünüz. Aslında o renk mavi. Siz yanlış görüyorsunuz. Bunu fark ettiğimizde montaja başlamıştık ve bu sahneleri tekrar çekecek zamanımız kalmamıştı. Piramitleri çektik. Uzay gemisini çektik. Bertem kartondan bir dilim karpuz yaptı ve onu uzay gemisinin üzerine yerleştirip onun da çekimini tamamladık. Herşey güzel... Ancak bir sorunumuz vardı: Uzay gemisi ve karpuz dilimi piramitlerden çok daha büyüktü. Uzay gemisi piramitlerle birlikte çekilmediği için bu sorunu montajda halledecektik ama karpuz ve piramitlerin bir arada göründükleri sahnede bu karpuzu kullanmamız mümkün değildi. Tüm objeleri hazırladıktan sonra, içeride keyifle televizyon seyreden Bertem'e bir karpuz dilimi daha yapması gerektiğini, ama bunun sadece 3 cm. Büyüklüğünde olması gerektiğini söylediğimizde sıraladığı karpuz fantezili küfürleri hayatımda ilk kez duyduğumdan eminim. :)

Kısa süren ikna çabalarımızın ardından Bertem'e 3 cm'lik karpuz dilimini de yaptırdık ve piramit sahnesi çekimlerinin büyük bölümü tamamlanmış oldu. Sadece piramitin uzay gemisine ateş edeceği kısım kalmıştı. Bu kısmı bilgisayarda mı yapsak, piramitin içine toplu iğneler mi soksak gibi düşünceler sırasında, buraya en uygun objenin kibrit çöpü olacağına karar verdik ve evde kibrit çöpü aramaya başladık. Uzun arayışlar sonunda çöp kutusunun içinde iki adet yanmış kibrit bulduk ve bu kibritleri kullanarak piramit çekimlerini tamamladık. (Bu arada size bir de sır vereyim: Aslında uzay gemisine Kefrens piramidi değil, Cheops piramidi ateş ediyor. :))



Şekil 4.

KUZU KELLE

Demoda piramitli bölümün sonunda R.W.O tarafından çizilmiş güzel bir pixel grafik vardı ve bunu andıran bir kare elde etmeliydik. Bir yaratığın kafası, bir kertenkele ve bir şişe! Şişe en kolayıydı. Bir oyuncak kertenkele buluruz, o da tamam. Peki ya yaratığın kafası? Uzun araştırmalar ve beyin fırtınaları sonucu birkaç saniyede bu yaratık kafasının bir kuzu kelle olabileceği konusunda Yatuyu ile hemfikir olduk. Bu engeli de çok kısa sürede aştığımızı sanıyorduk ama kazın ayağı hiç de öyle değildi. Bu tek bir kare görüntü, demoda bizi en çok uğraştıran kısım olacaktı...

Şekil 5.

Kuzu kellenin fiyatı hakkında hiçbir fikrimiz yoktu. Önce bir kasapa gidip fiyatını öğrenelim dedik ve acı haberi aldık. Kasap bize; "Zonguldak'ta küçükbaş hayvan kesimi yapılmıyor, et Ankara'dan geliyor, talep de olmadığı için kimse kelle getirmiyor, hiçbir yerde bulamazsınız" dediğinde kendisine inanmak istemedik. (Hayır, başka bir yazıya ışınlanmadınız, demo yapıyoruz burada.) Birkaç yere daha sorup yine aynı sözleri duyduk ve sonunda bir kasaptan, nispeten güzel bir bilgi alabildik. İstersek bize Ankara'dan kuzu kelle getirtebilirdi. Bunun için sipariş verip,

3 gün beklememiz yeterliydi. Derin bir oh(!) çekip kertenkele aramaya başladık...

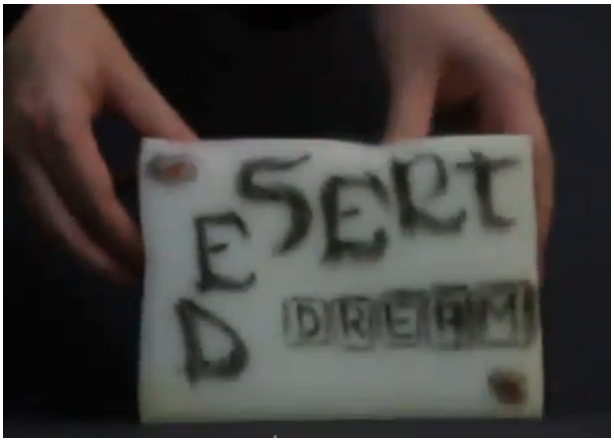
Şehrin en büyük oyuncakçısına gittik ve “kertenkele istiyoruz” dedik. Görevli arkadaş bizi hemen hayvanat reyonuna aldı. Eşekten su aygırına, deniz atından dinazora kadar her türlü hayvan mevcuttu ama kertenkele yoktu! Koca oyuncakçıda bir kertenkele bile yoktu! “Böyle rezalet görmedik, siz bu mesleği bırakın da gidin simit falan satın” şeklinde söylenerek mekandan ayrıldık ve diğer oyuncakçılara doğru yöneldik. Hiçbirinde yoktu. Şehirdeki oyuncakçıların hiçbirinde kertenkele yoktu! Kısa sürede bu şoku atlattık . Bu soruna bir çözüm bulmak zorundaydık. Yapılabilecek en güzel girişimi yaptık ve bu sorunu daha sonra çözüme kavuşturmak üzere erteledik ve diğer partlar üzerinde çalıştık!

Demonun ilerleyen bölümlerinde bir pixel grafik, zoom ve mirror efektlerinin olduğu bir part vardı. Bu parttaki pixel grafiğin çok benzeri Karadeniz Ereğli'de bulunan Herkül heykelinin bir parçasıydı. Herkül'ün öldürdüğü üç başlı canavar bu grafikteki canavara oldukça benziyordu ve bu canavarın bir fotoğrafını çekip, bu bölümde kullanmaya karar vermiştik. Bu fotoğrafı çekmek için Ereğli'ye gitmişken, kuzu kelle de soralım dedik. Ereğli Zonguldak'a 50 km. uzaklıktadır. Bu iş için 100 km. yol yaptık ve çok sayıda kasap dolaştık ama sonunda zafer bizim oldu. Kuzu kelleyi bulmuştuk!!! Canavarın fotoğrafını da çektik ve mutlu mesud bir şekilde Zonguldak'a döndük.

Kuzu kelleyi bulmuştuk ama henüz bir kertenkelemiz yoktu. Zonguldak'a döndüğümüzde Yatuyu ile birlikte her zaman alışveriş yaptığımız çiğ köftecimize uğradık ve o anda çiğ köftecinin hemen yanındaki oyuncakçıya daha önce girmediğimizi fark ettik. Oyuncakçıya girdik ve en güzelinden (veya çirkininden) bir kertenkeleye ihtiyacımız olduğunu söylediğimizde adam kararlı bir şekilde bizi kertenkele reyonuna davet etti. Kertenkele yoktu gerçi ama kertenkeleye “hık demiş burnundan düşmüş” şekilde (mecburen) benzettiğimiz bir timsah alarak oradan uzaklaştık. Bir engeli daha aşmanın mutluluğuyla o akşam çekimlere kaldığımız yerden devam ettik...

Akşam Squidward ile birlikte Bertem'e giderken, Bertem'in apartmanının önünde bir fıçı gördük. Fıçı ağzına kadar kül ile doluydu. Kaloriferci amca sağolsun bizim için tüm hazırlıkları yapmış. Kelleyi aldık küllerin üzerine yerleştirdik. Kertenkele görünümlü timsahımızı kellenin üzerine koyduk... Bize bir de şişe lazımdı. Sağa sola bakındım ve az ileride bir bira kutusu gördüm. Kullanmayı düşündüğümüz şişe bir Efes Pilsen şişesiydi ama kutu da pek tabii olurdu. Yaratık şişe efes değil de kutu efes içiyormuş. Olamaz mı yani! Yaptık, oldu. :)

SÜNGER JÖLE



Şekil 6.

Yatuyu'nun mesleği pastacılıktır. Bu işleri en iyi o bilir. Madem demoda bir jöle efekti var, gerçek jöle ile bu part pek tabi çözülecektir. Açıkçası ben başlangıçta böyle düşünüyordum ama bunu gerçekleştirmek pek de kolay değildi. Yatuyu'ya bu fikri açtığımda jöleyi o şekilde hareket ettirmenin imkansız olduğunu, bu girişimin sonunun hayal kırıklığı olacağını söylediğinde bu sahne için aklımda başka bir fikir yoktu. Ne yapacağımı bilmiyordum. Daha sonra sünger kullanmaya karar verdik ama süngeri kim düşündü hatırlamıyorum. Benim fikrim değildi, sadece onu biliyorum. Ama sonuç gerçekten güzel olmuştu. Sünger almaya gittiğimde bu süngerden iki adet alarak bir küp elde edebileceğimi ve küpün dört yüzüne desert dream logosu çizerek bu işin altından kalkabileceğimi sanıyordum ama dükkancı amcanın verdiği iki süngerin birbirinden farklı ebatlarda olduğunu evde fark edince işler değişti. Tek süngerin iki yüzüne Desert Dream logosu çizdim, ince yüzlere de Zomco yazarak durumu kurtarmaya çalıştım. Gelen tepkilere bakılırsa bu sahne hiç de fena olmamış. Bu kadar beğenileceğini beklemiyordum açıkçası... (Sahnedeki eller bana ait. Önce elleri yok edelim diye düşündük ama sonra vazgeçtik. Sanırım böyle daha iyi oldu.)

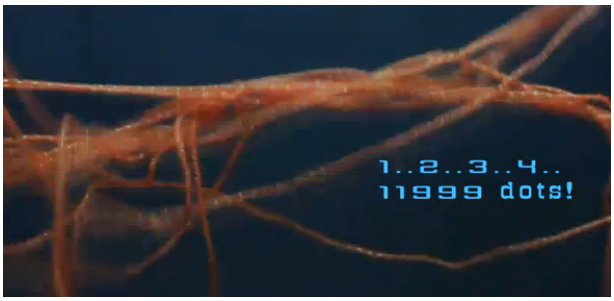
FAKE ZOOMER

Aslında bu part bu şekilde olmayacaktı. Bir kağıda teksti yazıp, kamera ile zoom in / zoom out yaparak bu sahne tamamlanacaktı, ancak yaptığımız ilk çekim istediğimiz sonucu vermedi. Bu partı tekrar çekmek için vaktimiz kalmayınca son dakikada bilgisayar üzerinde hazırladık. Biliyorum, bu şekilde berbat oldu...

11999 DOTS

Desert Dream'deki 12000 dots partını izlediğimde aklıma direkt olarak Öz Desert Dream'de yaptığımız uygulama geldi. Bunun için janjanlı yüne ihtiyacımız vardı. Ama sorun şuydu ki; janjanlı yünü yüncüden nasıl isteyeceğimi bilmiyordum. Kendi aramızda konuşurken bu yünün adı janjanlı yündü ama asıl adı kimbilir ne idi! Yüncüye gittim, kapıdan içeri girdiğimde istediğim şeyi nasıl ifade edeceğimi hâlâ bilmiyordum. “Şimdi bana şöyle bi yün lazım” dedim... (Derken de raflara bakıyorum o yünü görebilecek

miyim acaba diye. Görsem direkt, “aha bu” diyeceğim.) “Hani şöyle parlak parlak parıldayan bişeyler var üzerinde...”, “Nasıl anlatsam bilmem ki...” Kızcağız bana renk soruyor, ne örülecek falan diyor ama benim o janjanlı yünü görüp göstermem lazım, başka kurtuluşum yok. Tıkandım kaldım artık, söyleyecek bişey kalmadı; “janjanlı yün” dedim, “renk önemli değil, yeter ki parlak ve janjanlı olsun.”



Şekil 7.

Tezgahtar kız derdimi anladı. Dükkana girip başından “bana janjanlı yün verin” desem olacakmış yani. Bana yünün ismini de söyledi sonra ama aklımda kalmadı. Umarım bu efekti bir daha yapmam gerekmez. :)

5 DAİRE

Bu part en kötüsüydü. Daireler kartondan kesilecek ve misinalarla veya başka bir yöntemle hareket ettirilecekti ama son dakika kurbanı bölümlerden biri oldu. Bu part demoda olması gereken bi parttı ve kesmek istemedik. After Effects ile hazırlandı ve evet, iğrenç oldu. Squidward bu partı nasıl hazırladığını kendi bile bilmiyor. :)

Orijinal demoda bulunan birkaç partı da ya gerçek hayatta nasıl uygulayacağımızı bulamadığımız için, ya da zamanımız kalmadığı için kesmek zorunda kaldık. İki demo arasındaki yedi farkı bulma işini size bırakıyorum. :)

ELEKTRİKLİ TESTERE

Yuvarlak testere bıçağını satın almak için mağazaya girdik, bıçağın fiyatını gördük ve çıktık. “Kartondan daha güzel olur, evet” diyerek Bertem'e karton bıçak siparişi verdik. Bertem o sırada uyku moduna girmişti ve bize uçlarının bi kısmı doğuya, bi kısmı batıya bakan berbat bir bıçak hazırladı. Son çekim günümüzdü ve bu bıçağı kullanmaktan (Squidward'ın baskılarıyla) vazgeçtik. Squidward “ben bilgisayarda hazırlarım, aynısı olur” dediğinde sanırım ben de uyku moduna çoktan girmiştim. Kesinlikle gerçek bir bıçakla (olmadı en kötü karton bıçakla) çekilmesi gereken sahne bilgisayarda hazırlandı. Aslında orijinal demodakine çok yakın bir sonuç elde ettik ama olması gereken bu değildi. Yine azalan zamanın kurbanı olduk.

İĞNELİ TOP

Dikkat ederseniz bu part orijinalindekinden biraz farklıdır. Daha doğrusu Öz Desert Dream'de orijinal demodaki partın ikinci yarısını görürsünüz. İğneler topa batmıştır ve top dönmektedir...

Şekil 8.

Aslında bu bölümü bu şekilde çekmedik. İğneler tek tek topa saplandı, top her defasında bir miktar döndürüldü, iğneler tekrar saplandı, top döndürüldü ve tüm iğneler saplandıktan sonra top bir süre daha dönerek sahne tamamlandı. Ancak bunun montajı biraz zahmetliydi ve evet, yine vakit yoktu. Montaj gerektiren kısmı çıkarıp bu partın ikinci bölümünü kullandık ve demonun ilk bölümünü tamamladık.

DİSKET DEĞİŞİMİ

Disket grafiği Octopus/Zomco tarafından 1993 yılında çizilmiştir.

PART II – WELCOME

Şekil 9.

O kadar kolay göründüğüne bakmayın, orada üç kişi canla başla çalışıyor. Yatuyu kamerayı kullanırken, ben yazıyı kaydırıyorum, Squidward ise laser pointer ile ışık efektini yapıyor. :)

RENKLİ ÇUBUKLAR

Bu bölüm bizi çok uğraştırdı. En çok kafa patlattığımız ve son güne kadar nasıl yapacağımıza karar veremediğimiz bir bölümdü. Önce renkli kalemler kullanalım dedik. Kalemleri birbirine, aralarında boşluk kalacak şekilde yapıştıracak ve ortaya çıkan objeyi kamera karşısında döndürecektik. Yanlış hatırlamıyorsam 48 adet kalem kullanmamız gerekiyordu ve bu kalemleri birbirlerinden birkaç milimetre mesafeli halde birbirlerine yapıştırıp, düzgün ve sağlam bir obje haline getirip, sonra döndüre döndüre çekimini yapmanın çok da kolay olmayacağına kanaat getirip, başka fikirler üretmeye koyulduk. Üç adet şeffaf kaset kapağını birbirine yapıştırıp bir piramit yapalım, sonra da kapaklar üzerine renkli çubuklar yerleştirelim dedik, sonra bundan da vazgeçtik. Aklımda bile kalmayan bir iki saçma fikir daha ürettikten sonra, renkli çubukları printer çıktısı olarak alıp, kağıdı kartondan yaptığımız piramidin etrafına yapıştırmaya, kartonun iki yanından geçireceğimiz misina ile kartonu boşlukta tutmaya ve elimizle vurarak çevirmeye karar verdik. Sonuç o kadar güzel oldu ki, kimse bu bölümde ne olduğunu fark etmedi bile. :)

TAHTA PARÇALARI

O tahta parçaları gerçek olacaklardı, olmalıydılar ama maalesef gerçek değiller. Bu bölümü de son güne bıraktık ve dört adet tahta parçası temin edemedik. Karton mu kullansak diye düşündük, sonra vazgeçtik. Sigara paketlerini üst üste koyalım dedik, berbat bir fikir olduğunu düşünüp vazgeçtik. Squidward “ben bu işi çözerim” dedi ve yine bilgisayar üzerinde halletti. Fena olmadı aslında...

4800 STARS IN 2 BITPLANES

Yıldızlar kartondan kesildi. Ama zoom işi bilgisayar marifetiyle yapıldı. Yine Squidward'ın işi. Son geceye kalan tüm çekimler Squidward'ın gazabına uğradı desem yalan olmaz. Aslında ona bakarsanız tüm demo After Effects ile hazırlanmalıydı... Squidward Zomco ekibine bu iş için katıldı. Scene ile uzaktan yakından alakası olan biri değildir kendisi. Merakı da yok aslına bakarsanız. Bu demoyu neden gerçek hayatta uyguladığımızı ona anlatmayı başaramadık. Ona göre bu efektlerin gerçek hayatta uygulanması çok kötü ve komikti. Aslında bizim istediğimiz de tam olarak buydu. Fakat montajı yapan Squidward'dı, çekimler yetişmemişti ve bazı partlar bilgisayar üzerinde hazırlanmak zorundaydı. Mecbur kaldığımız için bazı bölümleri onun istediği gibi yaptık. Ama partideki gösterimde kahkaha alan bölümleri gördükten sonra Squidward bu demonun neden tamamen gerçek hayatta uygulanması gerektiğini anladı. :)

4800 Stars In 2 Bitplanes bölümünü biz 2 Stars In 4800 Bitplanes şeklinde uyguladık. Sonra yıldız sayısını dörtte bire, yani yarım yıldıza düşürüp sahneye, artık grubumuzun simgesi haline gelen mengeneyi çıkardık ve bu bölümü de tamamlamış olduk.

BIG BALLS

Arda'nın da partide söylediği gibi, burada tenis topu kullanmayabilirdik. Aynı espri iki kez yapılmış oldu. Başlangıçtaki düşüncemiz de farklı toplar kullanmaktan yanaydı ama burada tembellik ettik, kabul ediyorum.

DOT TUNNEL

Evet, bu bölümde stress halkası kullanacaktık. Yapılacak iş gayet basitti. Bir oyuncakçıya gidilecek ve stress halkası alınacaktı ama biz o anda bu oyuncağın adının stress halkası olduğunu bilmiyorduk.

Şekil 10.

Oyuncakçıya gittik, ve oyuncağı tarif ettik: “Hani şöyle bi elinden alıyosun, öbür eline koyuyosun, böyle halka halka bi oyuncak var akordeon gibi, var mı ondan sizde?” Neyse ki adam anladı. Yok dedi. Modası geçmiş artık o oyuncağın. Başka oyuncakçılara gittik, yine tarif ettik, yok, yok, yok... Oyuncakçılardan biri bize bu oyuncağın adının stress halkası olduğunu söylediğinde, “oh” dedik, tarif derdinden kurtulduk. Bi sonraki oyuncakçıya “stress halkası var mı” diye sorduğumuzda “o da ne olaki” şeklinde bir cevap alınca, yine tarif etmek zorunda kaldık tabi. Netice itibariyle oyuncağı bulamadık. Sonra yakın zamanda biryerlerde bu oyuncağı gördüğümü hatırladım. Evet, çok yakınlardaydı. Dayımın 7 yaşındaki kızı Elif'in böyle bir oyuncağı vardı. Ama Elif'ten en değersiz oyuncağını bile isteseniz o oyuncak kıymete binecekti. Görev oldukça zordu. Elif ikna edilecek, oyuncağı kısa bir süre için ödünç alınacaktı. Neyse ki bu işi çözmek için muhteşem bir şey icat edilmişti. Çikolata!!! Elif ikna edildi ve oyuncak ödünç alındı.

Bu bölümde stress halkasını üç kişi tutuyor. Oyuncağın boyu topu topu 15 cm. var, yok... Ama o hareketi vermek kolay değil tabi. :)

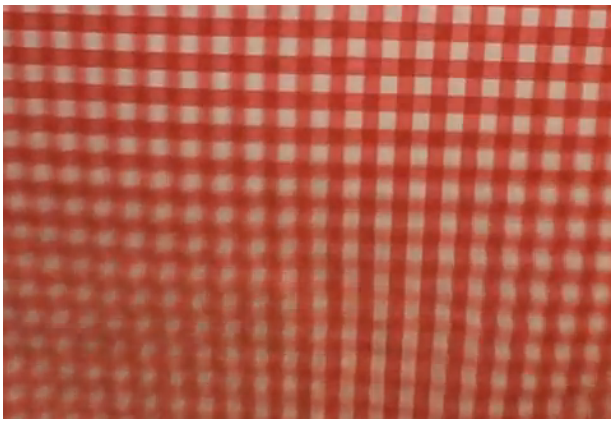
CANAVAR

Bilgisayar ile müdahale etmek şart olan bir part. Gerçek hayatta da çözüm mümkün ama sonuç kötü olacaktı. Bilgisayar üzerinde çözdük. Ama taaa Ereğli'ye gidip 10-15 kare fotoğrafını aldığımız canavarın en kötü fotoğrafını kullanan Squidward'ı bu vesileyle kınamadan geçemeyeceğim. :) (Montaj esnasında ben orada değildim)

SINEDOTS

Bu bölüm muhteşem olacaktı. En iddialı bölümlerden biriydi. Balıkçı ağı ile sinedots yapacaktık. Ama her yanı deniz olan bir balıkçı şehrinde balık ağı bulamadık. İpleri birbirine bağlayıp biz kendi balık ağımızı yapalım dedik, bu işi de son güne bırakınca yetiştiremedik. Bertem'in evinde bulduğum bir dantelvari “şey” ile çözdük. Bilgisayarda efekti verdik ve berbat bir sonuç aldık. Kimse ne olduğunu anlamadı bile.

MASA ÖRTÜSÜ



Şekil 11.

En bomba bölümlerden biri oldu. Bu projeyi hayata geçirmeye karar verdiğimizde Yatuyu evde eşi ile Desert Dream'i izliyormuş. Yatuyu'nun eşi Derya bu partı izlerken “burada masa örtüsü kullanın, ben size tam bu iş için uygun bir örtü vereceğim” demiş ve ertesi gün Yatuyu elinde masa örtüsü ile geldi. İlk çekimlerden biriydi. İzleyen herkesin en beğendiği bölümlerden oldu.

SON BAKIŞ

Bir demo parodisi yapmak daha önce kimsenin düşünmediği bir işti. Bir demoyu gerçek hayattaki objelerle yeniden oluşturmak daha önce denenmişti (bizim bundan da haberimiz yoktu) ama bunu bir parodi şeklinde sunmak ilk kez Zomco tarafından hayata geçirilen bir iş oldu. Daha iyi yapabilir miydik? Evet, çok daha iyi yapabilirdik. Biraz acemiliğimize geldi, biraz ciddiye almadık, zamanı iyi kullanamadık derken, ortaya böyle bir sonuç çıktı. Yine de bizim içimize sindi. İçimize sinmese zaten 7DX'te release etmeyi düşünmezdik. Herkes tarafından beğeni toplaması da bizi memnun etti. Hatta partiden sonra birkaç teklif bile aldık. Bizden bazı demoların parodisini yapmamızı isteyenler oldu. Şu an için ikinci bir demo parodisi planımız yok ama belli de olmaz. Bir anda gaza gelip bir parodi daha yapabiliriz. Kimbilir... :)

7dx 2010 Ürün İncelemeleri

Bilgem 'Nightlord' Çakır

Kürşad 'Hydrogen' Karamahmutoğlu

Metehan 'Spritus' Alter

Şekil 1.

Müzik Yarışması

Violin Edge

Nightlord: İlginç bir çalışma. Tam olarak ne yazacağımı belirlemekte zorlanıyorum. İlk dikkatimi çeken şey, başta ve sondaki rüzgar sesi efektini sevmediğim. Kompozisyon olarak fena bir parça değil. Kendini fazla tekrar eden yerler yok. Hayli dinamik bir akışı var. Bana göre parçanın cok güzel bir parça olmasını engelleyen iki büyük eksik var. Birincisi mix. Özellikle orta bölümde kullanılan bütün enstrümanlar mid frekanslarda birbirine karışıyor. Ayrıca bas gitar sesinin daha açık olması gerektiğini düşünüyorum. İkinci büyük problem ise parçanın akılda kalıcı ön planda dikkat çeken bir melodisi olmayışı. Ben her zaman melodi olmasını savunan bir insan değilim bazen parçalarda armonik-kontrapunta sekanslarını çok severim. Fakat burada açık şekilde önde melodi olarak yazılmış sekanslar var. O sekansların yeterince güçlü melodiler olmadığını düşünüyorum. Çok fazla değişiyorlar, ve çok uzun atlamalar yapıyorlar.

Dolayısıyla aslında bayağı dinamik ve enteresan alt yapısı olan bir parça, daha usta bir mix ve daha odaklı melodiler ile çok daha iyi olabilirdi.

Hydrogen: Açıkçası, yeni müzik programlarının yetenekleri konusunda bazen kafam karışıyor. Müziğin backroundu ile, melodileri sanki aynı kişi yapmamış izlenimi oluşuyor bazı yerlerde. Violin Edge'de bir kaos durumu hakim. Ana Violin sample'ı çok başarılı değil. Müziğin güzel kısımları 01:06'da giren bateri atraksiyonları ve 01:14'de giren tadında melodiler. 01:30'dan sonra melodiler biraz randomlaşıyor. 01:44'de giren piano partisyonu ise çok cılız ve öncesindeki coşkunluğu devam ettiremiyor. 02:30'da rastlansallık biraz karmaşaya dönüşüyor. Gene de, Leonard'da melodik yapı ve harmoni oluşturma konusunda genel bir hassasiyet ve çaba seziyorum ki, avrupadaki pek çok müzisyen, sadece tuhaf soundlarla, yüzlerce kere tekrar eden ambient müzikler yaparken, bu komposizyon oluşturma çabası takdire şayan. Sonuçsuz kalmayacağına inanıyorum.

SOS

Nightlord: Daha Trance havalı bir parça. SOS sinyalini kullanma fikri ilginçti. Yine alt yapısı gayet atmosferik. Ritm için kullanılan padler hoşuma gitti. Lakin Leonard'ın diğer parçası kadar dinamik bir parça değil bu. Tabi bu birazda electronic müzik janrasının daha tekrarlı olmasından kaynaklanıyor. Malesef Violin Edge'deki melodi problemi bu parçada da çok güçlü şekilde var. Bana göre Violin Edge, compoda SOS'in önünde yer almalıydı.

Her iki parçada da biraz acelecilik sezdiğimi söylemeliyim. Böyle oturulup bir akşamda 3 saatte yapılmış parça izlenimi var. Leonard arkadaşımıza naçizane önerim parçalara daha fazla zaman ayırması. Belki de bunları o da benim gibi partiye yetiştirmek için tam bitirmeden gönderdi bilemiyorum. Ama bence her iki parçanın da daha en az 3-4 saatlik işi var.

Hydrogen: SOS genel olarak tek bir tema üzerine kurulmuş bir müzik. Bol bol S.O.S efektleri müziğin üzerinde dolaşıyor. Hemen hemen tekrarlardan oluşan 1:30 sn'lik bölümün, kendi içerisinde 2 kez tekrar etmesi ile ortaya 2dk 54 snlik bir müzik çıkmış. Ancak genel değişkenlik çok yetersiz. 00:48'de giren melodiler, rutini bir ölçüde kırsa da, 01:03-01:14 arasındaki melodi oldukça iğreti duruyor, sanki bir an evvel 01:15'e ulaşmak için bir çaba var. Ve evet 01:15'de müzik hoş sound ediyor. Açıkçası müziği 1:43'de bitiyor farzediyorum. Fena bir müzik değil. Bir süre sonra kulak alışıyor.

Puppets' Show

Nightlord: Çok atmosferik ve usta bir girişle başlıyor parça. Derken 0:30 civarlarındaki garip pıtch bendler beni atmosferden kopardılar. Şu genel iki sesli arpej, bas ve ritm davullar çok başarılı. Öne giren keman melodisi de yine biraz akılda kalıcılık açısından zayıf olsa da yeterince iyi.

2:08 civarındaki transpose parçaya çok lezzetli bir dinamik getirmiş. 2:30 civarındaki geri dönüş de öyle. 2:50 civarında giren 2 sesli melodi parçanın en akılda kalıcı melodisi.

Tümüne bakınca gayet beğendiğim, güzel ve atmosferik bir çalışma olmuş. Mix'te daha iyi olabilecek şeyler var ama bu hali de baya iyi. İki sesli arpejin sağ ve sola dağıtılması güzel. Bas birazcık daha açık olabilir. En azından 1.50 civarında giren elektro gitar lead'den sonra zayıf kalıyor. Bası o bölümde biraz daha yükseltse daha iyi olurdu sanırım. Fakat seslerin ayrımı gayet iyi. Frekans slotlama başarılı. Müzikle ilgili bana göre iyileştirilebilecek son nokta parçanın son bölümündeki bateriler. Çok "bilgisayar baterisi" şeklinde kalıyorlar. Daha "human" olmaları gerekiyor bence. Bunun için vuruşların şiddetleri ve zamanlamaları daha dinamik yapılabilir.

Gayet güzel bir parça. Allamulax'ın ellerine sağlık.

Hydrogen: Puppet's show, tansiyonu oldukça iyi ayarlanmış bir giriş bölümü ile bizi karşılıyor. 00:30'a kadar olan bölüm, gerçekten çok başarılı. 00:29'da giren bilinçli detonelik ve 00:30'daki atonal bölüm de harika diyemesem de enteresan. 00:40'daki ana keman melodisi ve genel harmoni gerçekten çok güzel ve bence şarkının zirve noktası. 01:41'da, geri plandaki bass, hoş bir Jazz solosuna giriyor. ancak bu solo çok çabuk bitiyor, 5sn'den biraz daha uzun sürseydi çok daha anlamlı olabilirdi 01:46'da gene ana melodi, başka bir lead enstrüman eşliğinde giriyor. Buradaki lead sound biraz fazla belirsiz. Arka plandaki akış, çok uzun süre aynı devam ediyor ve hatta 02:06'daki transposizyon dahi, monotonluğu çok aşamıyor. 02:30'da gene aynı melodiye dönülmesi ile, bence oldukça başarılı bir şekilde başlayan bu müzik, biraz sıradanlaşıyor. 02:50'de yani son bölümde, başlangıçta keman ile çalınan melodinin, başka bir enstrümanla tekrar edilmesi, müzikal bütünlük açısından isabetli olmuş. Burada genel olarak soundun kuvvetlenmesi de şık olmuş. Genel olarak Puppet's show, biraz daha kısa olsaydı ve cazip öğeler biraz daha belirginleştirilseydi, daha da iyi olabilirdi. Ancak şu haliyle de gayet güzel bir müzik.

In the Morning

Nightlord: Slowhand'in dönüşü bence kelimenin tam anlamıyla aslan gibi oldu. Slowhand zaten sound olarak çok iyi bir SID müzisyeni. Ben asıl Rising'deki müziğe daha çok vuruldum ancak In the morning de gayet güzel bir parça. Bana göre tek eksiği bütün parçanın aynı ritm ve akor dizisi önünde devam etmesi. Sanırım kısa zamanda compoya yetiştirilmek için yapılmış olması yüzünden böyle bir durum var.

Fakat buna karşın compoda daha aşağıda yer alan parçaların neden önüne geçtiğinin çok basit bir nedeni var. Melodiler. Gayet leziz ve akılda kalıcı melodiler ile parça dinleyiciyle hemen bir bağ kuruyor.

Sağlam bir parça. Daha nicelerine diyorum.

Hydrogen: In the morning, Reggae ritimleri üzerine kurulmuş, hoş akorlara sahip, melankoli ile umut arasında giden ilginç bir müzik. İnsanı yormayan ve rahatlatan bir doğaya sahip. Keşke biraz daha fazla melodik öğe barındırsaydı. 00:14 , 00:26, parçanın backbone'unu oluşturan, en keyifli kısımları. Soundlar çok spesifik olmasa da, parça gayet iyi sound ediyor. Peki Slowhand'in 7dx 2010'da yayınlanan iki müziği arasından, Rising'mi, In the Morning'mi deseniz, kesinlikle Rising derim:)

Trapped

Hydrogen: Nightlord'u bu müziği çok duygulu ve güzel bir bestenin, malesef aynı güzellikte aranje edilememesinin, müziğin gerçek değerine ulaşmasının önünde engel teşkil edebileceğini örnekliyor. Genel komposizyon gayet başarılı. Melodi uzunlukları tadında. Müzik insanı sıkmayacak ölçüde değişkenlik içeriyor. Harmonik öğeler, özellikle bazı bölümlerde harika kullanılmış. 00.28 ile ağırlığı artan yaylılar, 00:38 ile gerçekten çok güzel tat alıyor. Ve 00:50'lerde giren gitar melodisiyle muazzam güçlü bir şekilde kavuşuyor. Gitar solosu, melodik anlamda gerçekten çok başarılı. 01:30'daki duraklamanın arından giren sert riffler ve arka planda yükselen üflemeliler, müziğe değişimi tam zamanında katıyor. Ardından gitgide sertleşen sound eşliğinde giren ikinci gitar solosu bizi bitişe bağlıyor. Ancak bu güzelliklere, müzikal kıyım diyebileceğim kayıt ve icra hataları açık bir darbe vuruyor.

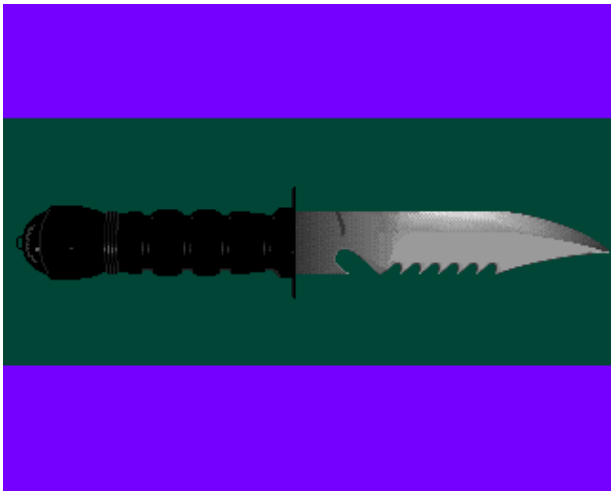
Arpej esnasında 00.07'deki elin slide ederken gitar teline sürtmesi kaynaklı ses, arpej boyunca loop ediyor. Harika başlayan lead gitarın 01:00'da hatalı-gecikmeli icrası sonucu oluşan küçük kaosun müziğin en duygusal noktasına vurduğu darbe. Gene aynı şekilde 01:10'da ki küçük kaos. Ara ara gerçekleşen belli belirsiz 1-2 hata. Bunlar bence kompozisyonun başarısını anlamayı güçleştiriyor, zira parçayı bu hali ile iyi algılayabilmek için, bir kaç kez dinlemeniz gerekiyor. Olumsuzluklara rağmen Trapped bir ara mp3 playerımda sürekli loop ediyordu ve hüzünlü bir modda dinlendiği zaman gerçekten sarsıcı bir melodik ve harmonik bir güç şarkıda mevcut.. Trapped kesinlikle rafine bir final versiyonu hakeden, karakter sahibi bir beste.

Pixel Grafik Yarışması

Spritus: Her yıl pixel grafik yarışmasının katılımcıları genelde aynı olurdu. Fakat 7dx 2010'da, Arcane/Glance'i hariç tutarsak 2 yeni grafiker katılım göstermiş. Üstelik kaliteli çalışmalarla. Bu yarışmada Turbo/Bronx, Hydrogen/Glance gibi kaliteli pixel artistleri göremediğimiz gibi maalesef ben de katılım gösteremedim. Partide bir ara Drey'in de bir pixel girişimini gözlemledim fakat yetiştiremediği için yarışmaya katılmadı. Yine de Drey gibi yetenekli bir scener'in bu girişimi beni heyecanlandırdı. Umarım bu alanda da güzel eserler vererek bize ne derece çetin bir rakip olduğunu gösterir diye temennide bulunarak yarışma ürünlerine geçiyorum.

Piçak

Hydrogen: Hoş bir piksel, fakat kompozisyon açısından biraz yalnız, çizilen objenin de kolay bir obje oluşu, resmin genel değerini düşürüyor.



Şekil 2.

Spritus: Octopus'un yarışmaya gönderdiği son çalışma olan bu

grafik görece diğerlerinden daha basit olmuş. Amiga'nın klasik grafik modunda 15 farklı renk kullanılarak çizilmiş. Grafikteki bıçağın siyah sapında görülen minik ışık oyunları ve metal parçada bulunan tarama geçişleri dışında açıkçası bahsedilecek pek bir tarafı olmayan bir çalışma. Yine de Octopus'un çizdiği diğer grafiklerden, onun ne kadar yetenekli olduğunu görebiliyoruz ve kendisinin bu yolda devam etmesini, scene'e böyle güzel katkılarda bulunmasını diliyorum.

Moganda

Nightlord: Savagery'nin başka birkaç çalışmasını daha önceden commodore.gen.tr forumunda görmüş ve beğenmiştim. O yüzden bu compodaki çalışmasını merakla bekliyordum.

Çok sevimli bir piksel işi. Bu çalışmayı karikatürizasyon ve tasarım açısından değerlendirirsek çok çok başarılı olduğunu düşünüyorum. Karakter tasarımı bence şahane. Bir elde rakı öbür elde tesbih gerçekten çok leziz. Ayrıca genel kompozisyon, arka plandaki dünyanın yuvarlak olması, karakterin yüz ifadesi falan hep alkışlanacak güzellikte.

Öte yandan malesef pikselleme tekniği olarak bazı zaaflar var. Öncelikle beğendiğim bazı diğer pikselleme detayları arasında, bıyıklardaki tonlama, rakı bardağındaki cam hissi, karakterin sağ dizindeki tonlama var. Ancak grafiğin büyük bölümünde hiç anti aliassing yapılmaması grafiği amatör gösteriyor. Böyle karikatürize flat tonlamalı çalışmalarda bile yüksek kontrastlı renk sınırlarında (mesela gökyüzü ve ay arasında veya gokyuzu ve beyaz atlet arasında vs) mutlaka anti aliasing yapılmalı. Mesela karakterin şortundaki kıvrımları yaparken yapılan antı aliasing gibi.

Bu kenarlarda daha smooth bir tonlama ve daha hareketli bir tarama ile compoda başa güreşebilecek bir iş olabilirdi. Bence bu haliyle de daha yukarıda olabilirdi. Savagery'ye tebrikler.

Şekil 3.

Hydrogen: İlginç bir karikatür ve başarılı bir uygulama. Özellikle renkler güzel kullanılmış. Atletin kıvrımları,paçalı donun deseni, gölgeler, backroundun resmi öne çıkartan gece tonları vs. Ancak pikselleme olarak başarılı olsa da , kompozisyonun biraz belirli bir beğeni grubuna hitap etmesi (ben bu tarz katikatürleri genelde es geçerim), resmin kitlelere ulaşımında sorun çıkarabilir:)

Spritus: Bu grafik, partide perdeye ilk yansıtıldığında insanlardan gülüşmelerin yükselmesine sebep olmuştu. Sanırım Savagery'nin amaçlarından biri de buydu. Bence yarışmada konsept açısından en başarılı eser bu grafikti. Ama maalesef resimdeki karakterin müstehcen görüntüsünden midir bilmem, hak ettiği yeri alamadı. Ben pixel çalışmalarında sanatsal öğelerin yanında eğlence öğelerinin de kullanılması taraftarıyım. Bu yüzden bu eser bana daha sempatik geldi. Grafiğin boyutu 320x256. Grafikte kullanılan unique renk sayısı ise 50. Octopus'un çizimleri gibi bu grafik de sanırım Amiga'da çizilmiş.

Grafikteki ayrıntılara geçecek olursak; öncelikle elinde rakı ve tespih bulunan, beyaz atletli, beyaz çoraplı, sivri burun ayakkabılı karakter, ciddi bir görünümden uzak karikatüristik bir karakter olsa da, sanırım toplumun bir kesimine eleştiri ve gönderme niteliği taşıyor. Bunun dışında yine karikatüristik bir biçimde oval çizilmiş taşlı zemin ve şehir silueti, resmin kalan kısmını dolduran ay, bulutlar ve ağaç gayet başarılı olmuş. Fakat bu grafikte de yine gözlerim şık renk geçişlerini, üzerinde özenle çalışılmış kaplamaları ve kırıksız smooth hatları aradı. Savagery'nin yine güzel çalışmalarını görmeyi umuyorum.

F16

Nightlord: Enteresan bir çalışma. Benim çok tarzım sayılmaz o yüzden yazacağım çok birşey yok. F16'nın formu çok ustaca yapılmış. Bulutlar ve güneş de öyle. Zemindeki tonlama da amaca uygun.

Duyduğum kadarıyla bu çalışma Octopus tarafından çook uzun yıllar önce yapılmış. Zaten 90'larda bir oyunun yükleme grafiği olabilecek nitelikte ve atmosferde bir çalışma. Belki Octopus tekrar piksel yapmaya döner.

Hydrogen: En sevdiğim tarzlardan biri idi, batan güneşin yarattığı kahverengi gökyüzünün önündeki siyah objeler. Ocp Artstudio'da 91 senedinde yanılmıyorsam, buna çok benzer pozda bir F16 piksellemiştim.

Oldukça atmosferik bir görüntü. Fakat hemen hemen hiç smooth (bazıları antialiasing olarak bilebilir) atılmamış piksel kenarlarına. Bu da bir Amiga resmine göre, fazla tırtıklı bir görüntü ortaya koyuyor.

Şekil 4.

Spritus: Octopus'un yarışmaya gönderdiği 3 çalışmasından ikincisi olan bu grafik 320x256 boyutunda olmasına rağmen kullanılan alan 206x179 ve toplam 14 farklı renkten ibaret. Octopus bu grafikte de makine/alet konseptini tercih etmiş. Grafik çalışmalarında konsept seçiminde daha dikkatli olunması gerektiğini yukarda belirtmiştim Bunun dışında grafiğin çizim tekniği, renklerin kullanımı gayet başarılı. Güneşin bulutlara yansıyan ışıkları ve uçağın siluet görünümü çok etkileyici. Ayrıca yeryüzü şekillerindeki ışık yansıması da hoş ve ilk grafikten farklı olarak tarama da kullanılmış. Ama maalesef yine anti-aliasing yok ve yine kırıklı hatlar var.

Hydrogen's C64

Nightlord: Tek kelime ile muhteşem bir piksel işi. Klavye tuşlarındaki inanılmaz piksel tekniği, monitör ve kasadaki gölge ve parlamalar inanılmaz usta işi. Ben parti boyunca Arcane ile bir iki kere muhabbet edebilmiştim, bir ara "neyse ben pikselime döneyim" diyerek kalktığında heyecanlanmıştım, "ne yapıyor acaba" diye.

Resmen makine ekrandan fırlayacak gibi görünüyor. Kontrast ile bu kadar ustaca oynanabilirdi.

Hala klavyeye baktığımda oradaki tuşları nasıl pikselliğine akıl sır erdiremiyorum. İnanılmaz bir ustalık.

Bu arada logo tasarımındaki güzelliği de es geçmeyelim. Özellikle a ve n harflerindeki simetri ve yeşil tonlardaki düşük kontraslı modernistik tarama çok hoşuma gitti.

Bu benim bu yarışmadaki favorimdi. Glance ürünü olduğu veya C64 ürünü olduğu için duygusal davranıyor olabilirim belki ama gerçekten inanılmaz ustalık fışkıran bir ürün olduğunu düşünüyorum. Arcane sen nasıl bir adamsın.

Hydrogen: Piksel tekniği ve renklendirme gerçekten kusursuz. Monitör ekranındaki reflection, ekran kenarlarındaki siyaha boşluklar vb. detaylar, tuş takımının kusursuz pixellenmesi, monitör ve bilgisayar üzerindeki yansımalar, ışık açıları vs. Top quality bir iş. Aşağıda bulunan Glance logosunu ise çok beğendim, hatları oldukça orjinal.

Arcane bu grafiği parti esnasında masanın üzerinde duran bir c64'e(Benim c64:)) bakarak bir kaç saat içinde çizdi ki, o ortamda bu işi çıkarabilmesi de gene takdire şayan kanımca. Grafiğin tek problemi ekrandaki boş alanlar.

C64 bitmap modunda ultimate realizm nasıl elde edilir sorusunun cevabını bu çalışmada bulabilirsiniz.

Şekil 5.

Spritus: Yarışmada 2. olan pixel grafiğimiz Glance grubunun emektar grafikeri Arcane'den. Bu grafik c64 platformunda multicolor renk modu kullanılarak joystick ile çizilmiş. Ayrıca grafiğin Arcane tarafından parti mekanında kısıtlı bir zaman içinde tamamlandığını da belirtmek gerek. Sanırım Arcane ne çizsem diye düşünürken karşısında kendisine gülümseyen Hydrogen'in c64 bilgisayarını fark edince hemen çizim girişimlerine başlamış olsa gerek. Grafikte c64'ün sınırlı renklerinin usta bir elden kullanımını görüyoruz. Bunun yanında bilgisayarın geometrisinde rahatsız edici herhangi bir perspektif hatası yok. Ayrıca grafiği besleyen ekstradan bir Glance logosu da mevcut. Her ne kadar şirin bir çalışma olsa da, kısıtlı zamanda çizildiği için Arcane gibi bir grafikere göre major bir çalışma olmadığını kabul etmek durumundayız. Bundan sonraki partilerde kendisinin daha ciddi çalışmalarını görmeyi umarak sıradaki grafik nasıl bir şeymiş bir bakalım.

Rally Team

Nightlord: Yine çok iyi bir çalışma, arabanın formu kusursuz. Işık ve gölgeler çok çok iyi. Bazı detaylar çok çarpıcı. Mesela farlardaki tonlama, tamponun sol alt bölümündeki gölgeleme falan harika. Tahminen bir fotograf model olarak kullanılmış olmalı.

Arabanın muhteşemliğine karşın arka plandaki yol ve çalılar bana biraz zayıf kalmış gibi geldi. Deluxe paint'te bir fırça yapılıp hızla boyanmış. Tabi yine bu da 90 başlarından bir çalışma olduğu için o zamanın estetiğine göre değerlendirirsek bu normal.

Sanırım compoda bu resmi birinci yapan faktör arabanın kusursuz formu ve ışıklandırması oldu. Daha önce de dediğim gibi

Octopus'un tekrar piksel yapmaya dönmesi dileğiyle.

Şekil 6.

Hydrogen: Güzel bir resim. Sanırsam İskender Atakan'ın, 1990'ların başında kullandığı Grup A Lancia Delta Integrale'si. Aracın pixellenmesi başarılı. Bazı bölgelerde smooth eksikliği göze çarparken, bazı bölgelerde de smooth başarılı şekilde uygulanmış. Aracın detayları oldukça hoş bir şekilde pixele aktarılmış. Ancak araş üzerinde hafif bir reflection, ışığın farklı açılardan gelmesinden kaynaklanan ton farkları vs. göz bunları arıyor biraz. Kokpit içerisindeki pilot kaskları güzel piksellenmiş ve mat cam hissiyatı başarı ile verilmiş. Gene aracın kendi üzerine ve yere düşürdüğü gölgeler dikkate değer, hoş detaylar. Background ise, daha çok spray ile oluşturulmuşa benziyor. Asfalt ve bitki hissiyatları ilkel düzeyde. Gene de resme fazladan güzellik katmasa da, olumsuz da etkilemiyor. Tatmin edici bir piksel çalışma.

Spritus: Yarışmanın 1. olan bu pixel eser, 320x256 boyutunda ve 27 farklı renge sahip. Konsept açısından biraz zayıf olsa da çizim kalitesi açısından güzel bir çalışma. Çevredeki yeşillikler ve asfalt yol derinlik hissini verebilmiş. Asfaltın 2 renkli deseni bir hayli hoşuma gitti. Çalılıklar da minimum renk sayısı kullanılarak güzel bir şekilde ölçüsü kaçmadan noise edilmiş. Otomobilin hatları da iyi çizilmiş ve tekerleklerdeki desenler, farlardaki yansımalar gibi güzel ayrıntılara yer verilmiş. Otomobilin içindeki pilotların siluetleri de bence grafiğin en iyi taraflarından biri. Grafiğe hoş bir etki bıraktığını kabul etmek gerekiyor. Bunun dışında çok belli etmese de ışık ve gölgeler de grafiğe baya olumlu etkilerde bulunmuş. Bence bu grafiğin tek eksik tarafı çizimlerdeki kırıklı görünümler ve taramaların yeterince kullanılmayışıdır. Octopus/Zomco partide bulunmadığı için kendisi ile konuşamadım; konuşmuş olsaydık kesinlikle bunları belirtirdim Özellikle çalışmalarında anti-aliasing kullanmasının önemi hakkında. Ayrıca tarama desenleri, geçişler ve daha orijinal konseptler üzerinde çalışmanın faziletleri hakkında da konuşmak isterdim. Kendisini bu güzel çalışmasından dolayı tebrik ediyorum.

İllüstrasyon Yarışması

Hydrogen: Her sene illustrasyon yarışması hakkında düşünürüm. Türkiye'de bunca yetenekli insan varken, böyle hoş ve underground bir ortamda düzenlenen illustrasyon yarışmasına neden bu kadar az katılımcı katılır diye. Bir scene partisinde illustrasyon yarışmasının gerekliliğine %100 inansam da, bu güne kadar gösterilen sınırlı katılım bunu sorgulamama neden olmadı değil. Ancak bu sene Caner Uyanık, gönderdiği güzel illustrasyonlarla, bu kategorinin boşuna olmadığını bana bir kez daha ispat etti. Umarım gelecek partilerde şahsen tanışırız.

Nightlord: Bence de bu üç eser bugüne kadar illüstrasyon dalında 7dx partilerine katılan en üzerinde uğraşılmış en iyi kaliteli ürünler oldu. Umarım seneye karşısına ona denk başka illustrasyonlar da çıkar.

Dragon

Nightlord: Bence compolardaki en ilginç ürünlerden biriydi bu. Çok başarılı karakterler var yine bu resimde. Ekrana böyle bir karede canlı, huyu suyu olan karakterler koyabilmek gerçekten yetenek istiyor ve resim sanatının bence en tanımsız ve büyülü yönlerinden biri bu.

Teknik olarak da dikkatimi çeken boyamada kullanılan enteresan texturelar ve büyük dragonun bazı bölümlerindeki blur kullanımı oldu. Çizgiler de çok stilistik.

Çok eğlenceli bir çalışma. Caner Uyanık arkadaşımızın eline sağlık.

Şekil 7.

Hydrogen: Çizim tekniği açısından gene hoş bir tutarlılık mevcut. Özellikle hareketlerin doğallığı, resmin en vurucu kısmı. Turkuaz, kızıl ve kahve tonların başarı ile kullanılması, harika

renk dengesi, başarılı espas. Ejderha tasarımları da ne kadar orjinal bilemiyorum ancak oldukça başarılılar. Ejderhaların yüzlerindeki mutluluk ifadesi oldukça güzel betimlenmiş. Detay ve minimalizm arasındaki dengeyi başarı ile kuran, hoş bir illustrasyon.

Siyam

Nightlord: Bu da yine çok enteresan texturelar kullanan bir çalışma. Ama yine dikkate ilk çarpan şey çalışmadaki karakterizasyon. Bas karakterin omzundaki dövmeden boynuna indirdiği gözlüklere kadar şahane bazı detaylar, arka plandaki dragonlar hep harika karakterizasyon örnekleri. Caner Uyanık arkadaşımızın gözle görülür bir “hikayelendirme” yeteneği olduğunu düşünüyorum.

Şekil 8.

Hydrogen: Gelecekte, yılan-hidra benzeri yaratıklarla savaşan siyam kedisi konsepti, oldukça ilginç. Dağılmış evde bulunan resimler ve eşyalar dikkat çekici güzel detaylar. Ancak genel komposizyon biraz dağınık. Arka plandaki Hidra başlarının aldığı fazla blur ile, ön plandaki objelerin keskinliği biraz dikkat dağıtıyor. Normalde resme derinlik katmak için yapılan bu hadise, burada olması gereken şekilde uygulanamamış. Işıklanma ise, biraz fazla belirsiz ve muallak, yer yer biraz kaba. Daha keskin bir ışıkla, daha dramatik, hoş bir komposizyon elde edilebilirdi.

Soldier

Nightlord: Önceki iki çalışmasından daha yoğun bir şekilde fırça tekniği kullanmış boyama için. Özellikle karakterin yüzündeki tonlamaya hayran oldum. Ayrıca be resimde kontrastların kullanımı çok güzel. Yüzdeki berrak tonlama ile vücudun geri kalanındaki daha bulanık boyamanın yarattığı kontrast da bence çok lezzetli. Yine hakim olarak soguk tonların hakim olduğu resimde yüz ve saçtaki sıcak renkler de çok güzel bir kontrast. Resimdeki genel ışık ve gölge yine çok başarılı

Hydrogen: Soğuk ve yüksek tepelerin semalarında seyahat eden bir zeplin içerisindeki(arkadaki afişten zeplin olduğunu çıkarıyoruz), sırtını sıcak yastığına (veya koltuk) yaslamış bir asker. Pencere pervazına kolunu dayamış. Dışarıda uçsuz bucaksız dağlardan oluşan harika bir manzara var. Üzerindeki kıyafetler, gözündeki gözlük onu soğuktan korumak için uygun işlevsellikte. Silahının sadece üst kısmı görünüyor.

Şekil 9.

Bu çizimi çok başarılı buldum. Özellikle detay seviyesi resmin genelinde çok iyi ayarlanmış (Bir tek arkadaki afiş bu durumu biraz bozuyor) Çizim tekniği resmin her bölgesinde kendini hissettiriyor. Aktörümüzün yüz ifadesi, sigara keyfinden biraz uzak olsa da, özellikle bacaklarının konumu ve bir elinin daima silahta olması vs. hoş ayrıntılar. Gene renkler ortamın soğukluğu içerisinde homojen bir şekilte dağılmış. Askerin burnunda soğuktan oluşan kırmızılık ve kızol saçlar , resmin renk dağılımındaki homojenliği başarı ile kırıyor ve heyecan katıyor. Kompozisyonun en güzel kısmı ise, pencereden giren ışıkla zengileşen mekan. Tepedeki borular, arkadaki yastık(koltuk), küçük klostrofobik ortamdan derya gibi açık alanlara bakan o pencere, o mekanda olmayı arzulatıyor insana. Bir resimden başka ne beklenebilir ki?

Oyun Yarışması

The Last Soldier

Nightlord: Daha önceden örneklerini çok gördüğüm bir oyun mekaniği (XBOX Live Arcade’de en çok satılan oyunlardan bazıları). Bu oyun mekaniğinin iyi bir implementasyonu The Last Soldier. Müzik ve ses efektleri başarılı. Grafikler iş görüyor. Tek problem oyun dengelemede. Bu haliyle oyunda mermilerin gücünün veya atış sıklığının zayıf kaldığını düşünüyorum. Diğer bir sorun daha ikinci seviyede düşmanların çok hızlanması.

Tabi bir diğer problem de oyunun bir başlangıç ekranı olma-

ması. Her oyunda bir başlangıç ekranı olmalı. Exe'yi çalıştırdığımız gibi üzerimize saldıran düşmanlarla karşılaşmamalıyız.

Ancak bütün bunlara rağmen biraz daha efor ile cilalı bitmiş bir oyun haline getirilirse gayet zevkli bir oyun olur.

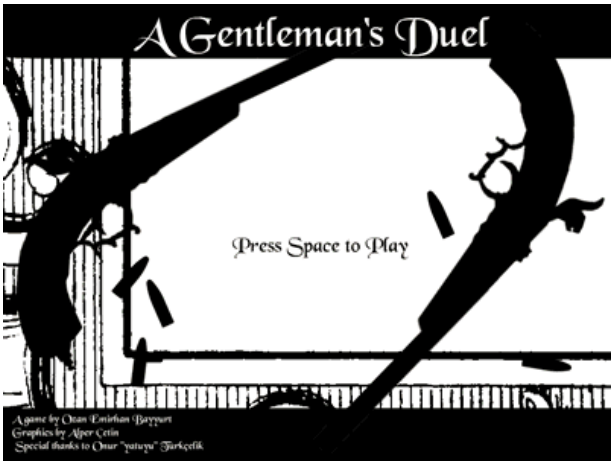
Şekil 10.

Hydrogen: Oyun Crimson Land klonu. Yanılmıyorsam parti mekanında yapıldı. Crimson Land'i izledim ancak pek fazla oynamadım. The Last Soldier, oynanabilirlik açısından, ortalamayı tuttursa da, öldürme-yoketme hissiyatı gibi konularda beni çok tatmin etmedi. Kurşunlarınızı zor görüyorsunuz. Vurulan tiplerin, uzuvları azalıyor ancak bu sizi görsel olarak çok etkilemiyor. Birkaç tip üst üste geldiğinde birden bire, 5-10 kişi birden ölüyor ve level bitiyor. Bu bir kaosa yol açıyor. Medikit gibi detayların atlanmaması güzel. Bu arada tekrar başlatma tuşunu bulamadım. Bir de score kaydedici olsa güzel olurdu.

A Gentleman's Duel

Nightlord: Bu oyunun mekanikleri ultra basit olmakla beraber, eğlenceli bir oyun. Fakat oyunun asıl çarpıcı tarafı bana göre grafikleri. Gerek başlangıç ekranının stilistik tasarımı gerek oyun ekranının güzelliği karşısında şapka çıkarıyorum.

Oyunun genel görüntüsü bana Monkey Island 3'teki banjo duellosu sahnesini hatırlattı.



Şekil 11.

Küçük de olsa cilalı ve bitmiş bir oyun. İşte oyun geliştirmeyle ilgilenen arkadaşların izlemesi gereken yol bu. Eğer vaktiniz az ise ve küçük bir oyun yapacaksanız, bu oyunu bitmemiş ve kalitesiz yapmak için sebep değil.

Bir başka deyişle, oyunun grafiklerinden, cilasından kısmayın. Oyun tasarımından kısın. Oyun mekaniği küçülsün. O küçük mekaniği de gerektiği kadar cilalayın. Gentleman's duel bunun için ideal bir örnek.

Hydrogen: Hoş bir fikrin hızlı bir uygulaması. Gentleman konsepti ve grafikler çok hoş. Animasyonlara ise, kesinlikle birkaç frame daha gerekirdi. 2 kişi aynı bilgisayarda oynamak zevkli olsa gerek zira ben, 2 elimi yarıştırdım. Bilgisayara karşı, çeşitli zorluklarda oynanış eklenseydi, tek tuş yerine shoryuken yapmak gibi çeşitli tuş kombinasyonları vs. olsaydı eminim daha da keyifli olurdu. Bu hali ile geliştirilmeye açık bir oyun. Umarım devam ederler.

Run Baby Run

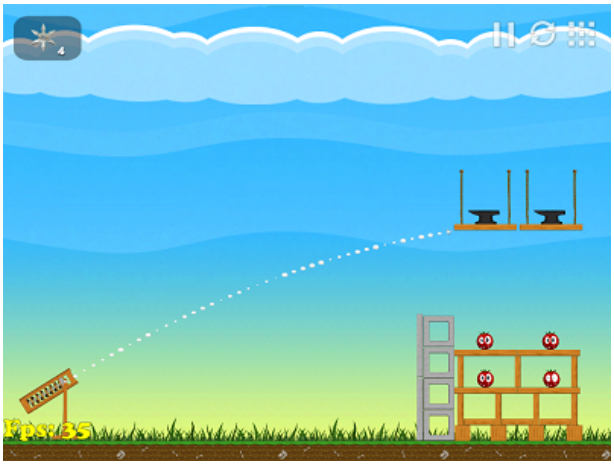
Nightlord: İlker Görkem bu yıl da güzel bir remake ile karşımızda. Geçen yılki River Raid remake'i orjinal oyunun verdiği oynanış hissinin aynısını verdiği için çok dikkatimi vermişti. Bu yıl yaptığı Run Baby Run ile ilgili benzer bir yorumu yapabilecek tecrübeye sahip değilim. Nitekim bu oyunu zamanında Speccy'de oynamışlığım yok.

Lakin burada bir anlığına içimdeki C64 canavarını serbest bırakacağım uyarıyorum.

Abi bunun orjinali ne kadar kötü bi oyunmuş böyle yahu. Bu Speccy'nin kabus grafikleri, ancak böyle zavallı oyun mekaniklerine izin vermiş herhalde. Yazık bütün Speccy kullanıcılarına. Sarı siyah abi. Olur mu ya :)

Evet yeniden c64 canavarını zaptettikten sonra devam edelim. Sanırım İlker'in en büyük şanssızlığı remake'ini yaptığı oyunun orjinalinin çok enteresan veya ikonik (River Raid gibi) bir oyun olmayışı oldu. Ee işte speccy fanatiği Ref'in ipiyle kuyuya inersen böyle olur İlker (heheh selam Ref naber)

Şekil 12.



Şekil 13.

Tabi belki de bu zamanında İlker'in çok sevmiş olduğu ve yapmak istediği bir oyun olabilir, bilemiyorum. Ama parti compolarında remake oyunla katılırken bence daha tribüne oynanmalı. Daha çok kişinin bilme ihtimali olan oyunlar yapılmalı. İhtimalen Atari veya C64 oyunları bu konuda en iyi adaylar.

Bunların dışında oyunun implementasyonu, ses efektleri, cilası tamamlanmış herşeyiyle bitmiş bir ürün olması ile benden çok büyük puan aldı. Ellerine sağlık İlker.

Hydrogen: Spectrum versiyonunu oynamamış biri olarak oldukça eğlenceli bir oyun olduğunu söyleyebilirim. Lucas Arts'ın Monkey Island'da yaptığının bir benzeri, burada yapılmış. Tek tuşa basarak oyundaki herhangi bir zamanda eski ve yeni grafikler(yeni grafikerler oyuna özel çizilmiş) arasında geçiş yapabiliyorsunuz. Bu özellik çok güzel düşünülmüş. Bendeki flash versiyonunun tuhaflığından olsa gerek, oyundaki sağa,sola dönme tuşlarını bayağı bir aradım. Bulduğumda ise, tam oyuna dalacakken birkaç yerde, oyun başa dönmeye başladı. Birkaç kere baştan başladım. Run Baby Run, hoş bir oyun ve güzel bir remake olmuş. Oynanabilirlik de oldukça başarılı. Spectrum sahipleri aslında bu oyun için çok daha fazla şey söyleyebilirler.

GMO

Nightlord: Bu oyunu sadece parti esnasında gördüğüm kadarıyla değerlendirebileceğim. Nitekim (tabii ki :)) Ipad sahibi değilim.

Oyunda dikkatimi çeken şey, oyunun ne kadar cilalanmış olduğu idi. Swift Development ekibinin ellerine sağlık. Oyundaki fizik, grafikler, ve ses efektleri çok başarılı idi.

Tabi böyle bir oyunda en önemli olay aslında dokunma arayüzünün hissi ve akışkanlığı. Onun hakkında da ancak bir Ipad ele alıp oynayarak yorum yazılabilir. Malesef o imkanım olmadığı için bu konuda yorum yapamıyorum.

Sonuçta çok cilalı ve bitmiş olması ile benden yüksek puan alan bir çalışma oldu GMO.

Hydrogen: Bu oyun, grafikleri ile, oynanışı ile, üzerinde bayağı uğraşılmış ve cilalanmış bir ürün. Parti mekanında izlediğim kadarıyla, fiziği olsun, grafikleri olsun, oldukça kaliteli bir katılımdı. Ancak maalesef Ipad için olduğundan dolayı oynayamadım ve hakkında detaylı yorum yapamıyorum.

Köy Korucusu

Nightlord: Bu oyunu önce makinemde çalıştıramadım. Ardından iki monitörden birini iptal edince çalıştı. Bu oyun enteresan bir oyun. Şöyle ki alt yapısı teknolojik olarak bayağı detaylı. Yani 3d ortam yapılmış. Animasyonlar, frame rate, oyun kontrolleri falan gayet güzel. Fakat oyun çok kısa sürede bitip tadı insanın damağında kalıyor.

Başlangıç ekranındaki müziği de çok beğendim. Orada biraz daha detaylı bilgi verilse daha iyi olurmuş. Geberik kanı tam olarak nedir, insan sayısı ne demek. Nasıl level atlanıyor, süre ne kadar vs bir sürü soru havada kalıyor. Ancak bu ihtimalen partiye yetiştirilmek için yarım bırakılmış. Bitmiş halini görmek ve oynamak isterim.

GMO ve Run Baby Run kadar cilalı olmasa da teknik olarak daha dikkat çekici olduğu için partide o iki oyunun önüne geçmiş olduğunu sanıyorum. Benim oyum Korucu ve GMO için (Korucu tam bitmemiş hissi verdiği için) eşit olurdu.

Fakat oyun oynanışı gayet güzel. Bu bölümün kodunu beğendim. Burada beni rahatsız eden tek birşey oldu. O da tüfeğin ateş ettikten sonraki tepmesi. Bu tepme zank diye sabit piksel yukarıya kayma şeklinde oluyor. Bunun yerine hızlı ama anime edilmiş bir tepme daha iyi olurdu.

Oyunun bitmiş haline eklenebilecek çok güzel fikirler geliyor aklıma ama şimdi burayı kalabalık etmeyelim.

Dolayısıyla bence rahat bir birincilik alacakken bitmediği için birinciliği almakta zorlandı bu oyun. Eline sağlık Infect.

Hydrogen: Bu oyunu, bana Beach Head'i hatırlatmasından olsa gerek, keyif alarak oynadım. Ancak skor sistemi muhtemelen hatalı (500'ü geçmeme rağmen 500 olarak yazıyor skorumu) dolayısıyla bir süre sonra skor motivasyonu kayboldu ve oyunu bıraktım.

Köye yaklaşan Zombie'leri engellemeye çalıştığımız oyunda, Zombie'ler kafalarından vurularak ortadan kaldırılabiliyorlar. Bir tüfeğimiz var ve bu tüfeğin dürbün gibi kullanışlı ve gece görüş gibi, eğlenceli ama kullanışsız özellikleri var. Köyün merkezindeki üç tepeye hakim küçük bir kulübede, her üç pencereye yürüyerek, içinde bulunduğumuz köye ulaşmaya çalışan Zombie'leri temizlemek eğlenceli. Müzik ortama atmosfer katarken, tüfeğin geri tepmesi falan güzel ayrıntılar. Grafikler vasat ancak oynanışa çok negatif etki etmiyorlar (Modellerde muhtemelen kafa tanımlanmamış. Ağırlık merkezinin belirli bir miktar üstüne bakıyor. Dolayısıyla, yan dönen öne eğilmis modellerde, kafa değil, biraz gerisine nişan almak gerekiyor)

Şekil 14.

Bir Zombie'nin köye tam olarak girdiği noktayı, Zombie'lerin ne ara köye girdiğini vs. tam olarak anlayamadım. Bunlar açık olarak belirtilseydi daha büyük heyecan yaratacaklardı. Geberik kanı gibi kötü Türkçe kullanımları, bana çok esprili gelmedi.

Gene bir olumsuz nokta, Zombieler kafadan ölürler tamam ama vücudun diğer kısımları vurulduğunda hiçbir tepki gelmemesi, ilk oyunu oynayanları, oyunun hatalı olduğunu düşünmeye itebilir. (Ben sürekli Headshot attığımdan, bunun çok geç farkında vardım eheheheh)

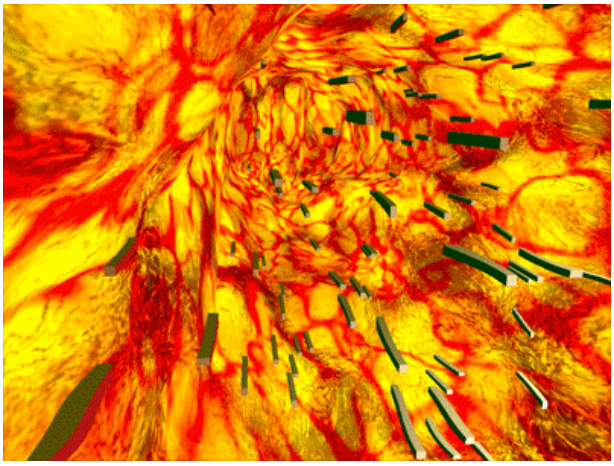
Sonuçta, hataları temizlendiği takdirde, başarılı ve eğlenceli bir oyun olağını söyleyebilirim.

Demo Yarışması

X2010

Nightlord: Infect bu yıl da yeni bir demo ile karşımızda. Bu demo beş bölümden oluşuyor. Bazı çok basit hatalardan dolayı potansiyelinin tamamına erişemeyen bir demo olsa da yine de partiye artı değer sağlayan bir ürün.

Önce demodaki beğendiğim şeylerden bahsedeyim. İlk effekteki tünel duvarlarındaki texture etkileşimleri iyi bir efektti. İkinci bölümdeki çöp adam skinning açısından her coderın geçmesi gereken bir nokta olduğu için olumlu (ki zaten Infect'in skinning yapabildiğini köy korucusu oyunundan da görebiliyoruz). Dördüncü bölümdeki tünel demoda post processing eksikliğinin göze batmadığı tek yer. Ve beşinci bölümdeki texture içine boyama efekti de güzel olmuş.



Şekil 15.

Fakat bu güzellikleri, bazı basit hatalar heba ediyor. Bunların birkaçını sayarak Infect'in sonraki demolarına yardımcı olmasını umuyorum.

1. Başlangıçta müzik ve efektin senkronize başlamaması.
2. İlk bölümde kameranın yeri ve kameranın front clip plane'ini keserek gelen prizmalar. Bu problem 4. Bölümde de var. Sahnedeki hiçbir geometri kameranın front clip plane'i ile hiçbir zaman kesişmemeli. Aniden geometrinin içinin açıldığını görmemeliyiz. Ve 3 boyutlu sahnelerde kameranın pek durağan olmamasında da fayda var
3. İkinci bölümdeki çöp adam animasyonunun sürekli olmaması. Burada sürekli bir walk-cycle kullanılsa çok daha iyi olurdu. Burada zemindeki yeşil/mor dama ve sütunlardaki yeşil sarı altıgenler, aceleyle bir grafikerin yardımının gerektiğini gösteriyor.
4. Dördüncü bölüm bence demonun en iyi bölümü. Burada sadece iki problem var. Bazen halkaların içinden keçerken kamera halkayı clip ediyor daha önce de söylediğim gibi. Birde bölümün ortasında tam kamera target'ın yanından geçip arkaya dönüyor kamera. O noktadaki dönüş çok haşin. Kamera targeti da yavaşça oynatıp o dönüş daha yavaş yapılmalıydı
5. Bitişte demonun paldırt diye bitmesi. En azından müziği fade out edip görüntüyü de siyaha fade ederek yumuşak bir şekilde bitmeliydi.

Bu yazdığım maddeleri yapmak sanırım Infect'in en fazla bir saatini alır. Ben bu demoya oylamada 6 verdim galiba. Bu dediklerim yapılsa 8 verirdim. Bu kadar basit düzeltmeler bazen bir demonun kalitesini çok farkettirebilir. Tamamen aynı kod, artı bir grafiker tarafından tasarlanmış texturelar, kamera hareketleri artı yukarıda bahsettiğim birkaç basit düzeltme ile bu demo bu yarışmayı rahat rahat kazanabilirdi.

Yine de gayet iyi bir demo ve Infect'in bundan önce yaptığı iki demodan da daha iyi. Parti raporunda daha önce yazdığım gibi, Infect'in artık fiziksel olarak partilere gelip scenerlarla tanışması lazım. Parti havasını soluması oradakilerle muhabbet etmesi ve bol bol demo seyretmesi lazım. Şu an Infect'in bana göre eksiği artık "Teknik" değil "Demo Vizyonu". Bu demo vizyonu da ancak bol demo seyredip bol bol başka scenerlarla etkileşerek geliştirilir.

Infect'in bir sonraki demosunu heyecanla bekliyorum.

Hydrogen: Evet, son dönemin en umut vadeden PC coder'ı Paradox, yeni demosu ile karşımızda. Bu demonun hemen her ekranında hem başarılı hem başarısız olaylar var. Bunun açık nedeninin, grafiker veya tasarımcı eksikliği olduğunu düşünüyorum. Efektleri incelerken olumlu-olumsuz her şeye değinmeye çalışacağım.

Açılışta, bir mağara-tünel ve bu mağaranın yüzeyinden kayan texturevari bir efekt var. İçerisinde de dikdörtgen prizma çubuklar ilerliyor. Güzel olan, kayan texture efekti, gerçekten bir demo görselliği taşıyor. Keyif verici bir efekt. Ancak tünel içerisinde burun istikametinde ilerleyen dikdörtgen prizmalar, genel görselliğe çok bir şey katmıyor. Belki yumuşak ve birbirinden farklı spin hareketleri ile, uzay boşluğunda, ağır ağır hareket etmeleri, efekti güzelleştirebilirdi. Bir sonraki ekran ise, tartışmasız demoyu geriye götüren bir ekran. Burada oldukça hatalı bir çöpadam animasyonu mevcut. Seçilen renkler vs. genel olarak görüntü oldukça başarısız. Bu ekran komple çıkarılsaydı, demonun geneli üzerine çok daha olumlu olurdu diye düşünüyorum. Ardından gelen ışık kaynağı kürelerin bulunduğu ekran, başlangıçta zor anlaşılıyor. Işığın yayılımı tam olarak doğru olmasa da göz alıştıktan sonra, fena görünmeyen bir efekt.

Gene bir tünel efekti ile devam eden demoda, efektin müziğin değişimi ile beraber gelmesi, hoş bir tat bırakmış. Burada, tünel zeminine oturmayan küpler, şekil şemali destekleyen torusların aksine, görüntüye zarar verse de, ışık geçişi olaya atmosfer katıyor ve durumu kurtarıyor. Son part ise, açık ara favorim. Landscape üzerinde ilerleyen renkler, gerçekten görsel olarak tatmin edici. Ve demonun, güzel bir efektle bitmesi isabetli oluyor. Genel olarak bütün partlarda, bir tasarım-grafik eksikliği var. Müziklerin de demoya özel yapılmamış olması bir dezavantaj. Ayrıca, tam olarak güzel görünmeyen, tatmin edici olmayan görüntüleri (Çöp adam gibi) demoya hiç almamaları da gene olumlu olurdu.İngilizce metin yazılması çok isabetli bir seçim, scene uluslararası bir oluşum olduğundan, grubun uluslararası alanda daha iyi tanınmasına yardımcı olur, ancak malesef, ingilizce çeviriler oldukça hatalı ve bu da tanıtımı iyi yönde değil, kötü yönde yapar. Eğer demoya ingilizce metin yazılacaksa, bunun elden geldiğince hatasız olmasına çok önem verilmeli. Eğer Paradox kendisine sağlam bir tasarımcı-artist ve müzisyen bulabilirse beraber çok güzel işler yapabilirler. Ortada gerçekten bir emek var ancak, bu öğeler olmadan tam olarak değer kazanamıyor.

Spritus: Gelelim başka bir Windows demosu olan X2010'a. Demoscene'in yeni sayılabilecek yüzlerinden biri olan Infect (a.k.a. Paradox) gerçekten yapıtlarıyla ve gayreti ile gelecek vadeden yetenekli bir coder. Scene'mizin Infect gibi üreten yeni nesil scenerlara ihtiyacı var. Ama daha önce de birçok tecrübeli scener ile olan konuşmalarımda da belirttiğim gibi Infect, bir grup içinde yer almadığı için, dizayn, grafik, kurgu gibi konularda yol gösterecek scenerlardan mahrum kalıyor. Bu yüzden de programladığı demolar, her ne kadar göz alıcı efektler içerse de partlar kopuk ve birbiriyle ilgisiz görünüyor. İzleyiciye kurgu / konsept açısından hiçbir şey izletemediği gibi demolarda çok önem arz eden geçiş efektleri neredeyse hiç bulunmuyor. Bu da demonun izleyicisinde bir tatminsizlik tadı bırakıyor. Kısacası Infect'in ya iyi grafiker ve tasarımcıların bulunduğu bir grupta yer alması gerekiyor, ya da kendisini tasarım konsept konusunda geliştirmesi gerekiyor. Aksi halde böyle değerli bir yetenek potansiyelini gösteremeden zaman geçip gidecek.

Gelelim demoya : Demomuz deforme olan bir tunnel içinde başlıyor. Tünelde yine deforme olan dikdörtgen prizmalar kameraya eşlik ediyor. Bir taraftan da tünel duvarlarındaki yansımaların değişimini izliyoruz. Bu efekt gayet güzel düşünülmüş, hoş bir seyirlik. Ama kamera çok statik. Biraz daha dinamizm katılabileceğini düşünüyorum. Bir süre sonra, sıradaki efekte, arada herhangi bir geçiş efekti olmadan geçiyoruz. En azından bir fadeout / fadein yapılabilirdi.

Sıradaki sahnemizde bir çöp adam animasyonu var. Çöp adam parlak bir zemin üzerinde sağlı sollu sütunların arasından ilerlerken etraftan küpler uçuşup gidiyor. Bence güzel senaryo edilmiş bir part. Ama yine statik ilerleyen kamera ve animasyondaki bug seyir zevkini biraz kaçırıyor. Bir süre sonra kameramız yaptığı hatanın farkına varmış gibi çöp adamın etrafında gezinmeye başlıyor. Daha sonra bu sahnenin ekranda bölünerek tile edilmiş bir şeklini izliyoruz ve maalesef yine geçişler yok.

Sonraki part petek deseniyle kaplanmış duvarların olduğu karanlık bir mekanda cereyan ediyor. Kameramız mekanda gezinirken duvarlara yuvarlak ışık huzmeleri gönderen topcuklar eşlik ediyor. Yine modern demolarda örneklerini gördüğümüz güzel bir efekt. Kaplamalardaki desenleri başarılı bulduğumu itiraf etmeliyim, yerinde ve güzel kullanılmış. Ayrıca efekte scanline'lı nostaljik bilgisayar ekranı havası katmış.

Bu bölümden sonra yine bir tünel efekti ile karşı karşıyayız. Tünelde ilerlerken değişik objelerle karşılaşıyoruz. Bir taraftan da görüntünün motion blur yapması hoş olmuş. Son partta ise bir terrain'in kaplaması üzerine yapılan basit boyama işlemlerinin olduğu bir efekt ile demo bitiyor.

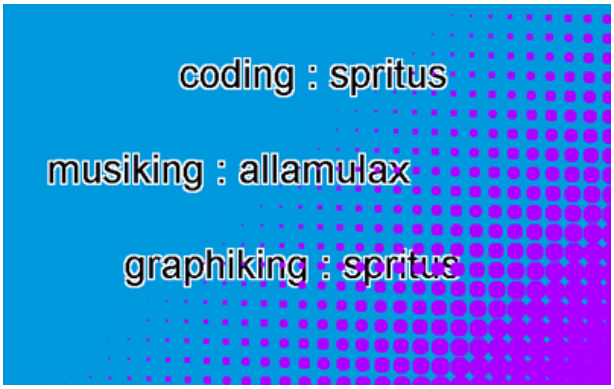
Bundan sonraki partilerde Infect'in gerçekten potansiyelini gösterdiği daha derli toplu demolar görmek en büyük temennim.

NKEOKK

Nightlord: Resident sonunda bir demo ile geri döndü. Bu zaten kendi başına sevinmek için yeterli sebep. Bunun yanında demonun kendisi de küçük, iddiasız ama temiz bir demo. Ne yazık ki demoyu Win7 üzerinde çalıştıramadım. Dalgalanan küplerden sonra hata verip çıkıyor. O yüzden partide gördüğüm kadarıyla

yorumlayabileceğim.

Öncelikle demin X2010'da daha iyi olabileceğini söylediğim ne vardıysa, onlar NKEOKK'de gayet iyi. Genel olarak renkler, zamanlamalar, akış, vs. Spritus'un tecrübesi sayesinde baştan iyi olmuş zaten. Demoda genel olarak hakim olan old skool tat da bütün efektlerce destekleniyor.



Şekil 16.

Efektlerin kodlanışı ve sunuşunda da herhangi bir pürüz yok. Dolayısıyla efektlerin zorlukları X2010'dan daha yüksek olmasa da sunuştaki hatasızlık yüzünden o demonun önüne geçti.

Demonun müziği benim için biraz problemliydi. Tam olarak demonun genel havasına uymadığını düsünüyorum. Demo daha oldskool bir demo, müzik ise daha modern experimental bir havada. Müzik daha old skool tekno veya en azından daha hızlı ritmli ve daha "loud" olsa demoyu daha iyi tamamlardı diye düşünüyorum.

Ancak Spritus ve Allamulax güzel bir ikili olmuşlar. Bu ikiliye bir de Tesla katılsa ne güzel olur diye içimden geçiriyorum.

Hydrogen: Uzun süren suskunluğun ardından, en nihayetinde bir Resident demosu ile karşılaşmanın mutluluğunu yaşıyoruz. Spritus'un ciddi anlamda programcılığa dönüş yapması, olaya extradan heyecan katıyor. Açılıştaki credits kısmı kendine has bir stile sahip ve oldukça hoş görünüyor. İlk efekt, voxel benzeri bir efekt olduğunu tahmin ettiğim, dalgalanan küplerden oluşan bir havuz. Oldukça sağlam bir görselliğe sahip olan bu partta, bence arkada dönen checkerboard environment, çok fazla dikkat dağıtıyor ve efekte kanalize olmayı zorlaştırıyor. Onun dışında demonun ve belki de 7dx 2010'un en güzel efektinin ekranda bir süre daha kalmasını istiyor gönül. Bir sonraki efekt, 3d küreler. Heaven Seven'da vs. gördüğümüz bilinen güzel bölümlerden biri olan bu efekt NKEOKK'da da çok hoş kullanılmış. 3d kürelerin hoş bir fake ile, bulutun arkasından çıkıyor izlenimi uyandırması, efektin etkisin oldukça güçlü hale getiriyor, derinlik katıyor. Arkaplandaki bulut resimleri de oldukça başarılı.

Greeting partına geldiğimizde... Burada da 3d uzayda rotate eden 2d sprite küreler var. Efekt başarılı. Ancak greetslerde, isim geçişlerine özenilmemiş. İsimler ekrana gelirken en basitinden bir fade vs. olmaması, biraz çiğ bir görüntü sunuyor. Ardından demo no-signal ekranı ile kesiliyor ve...

Bir sonraki partta 3d bir mağarada dolanıyoruz. Altıgen prizma ve bilimum prizmadan, daha manalı objelerle hoş olabilecek bir part ancak şu anki hali ile, mağara oldukça boş görünüyor. Efekt çok uzun tutulmuş, demonun pat diye birdenbire bitmesi fr, gene bu bölümün demoya etkisini oldukça olumsuz etkilemiş.

Demo grafik olarak bizlere bir şey sunmuyor. Coder'ın aynı zamanda Resident'in 1 nolu grafikeri olması bazı efektlerin görselliğinde fazlası ile göze çarpsa da, mevzubahis olan Spritus olduğunda, her ekranda sağlam bir grafik, logo veya grafiklerin dahil olabileceği efektler vs. bekliyor insan. Ancak gerçekten demoda hemen hemen hiç grafik olmaması bende biraz hayal kırıklığı yarattı.

Müzik olarak ele alırsak... Açılıştaki soundlar başarılı. Ancak genel olarak müzik çok fazla aynı spektrumda gidiyor. Volumun arttığı, melodinin başkalaştığı bir patlama noktası yok. Moody yapı müziğin ve demonun geneline yayılıyor. Bazı melodilerle müzik yer yer renkleniyor. Ancak bu uzun sürmüyor ve bir süre sonra aynı karamsar yapıya dönülüyor. Lead enstrümanların soundları çok belirsiz, bu da zaten ambient yapıda olan melodilerin bass ve crosslar arasında kaybolmasını sağlıyor. Demo ile müzik senkronizasyonunun da çok başarılı olmayışı, durumu olumsuz etkiliyor.

Kısacası, NKEOKK'da genel bir acele göze çarpıyor. Çok kolay şekilde düzeltilebilecek birkaç eksik nokta var. Eğer bu eksiklikler olmasaydı, çok daha başarılı bir ürün olabilirdi.Demonun yalnızca windowed çalışması ve bir takım hatalar içermesi de gene bir dezavantaj. Olumsuzluklarına rağmen bir Resident demosu izlemek her zaman güzeldir. Ve bu demonun, yeni gelecek harika stuffların bir habercisi olduğunu düşünüyorum.

Rising

Nightlord: Return grubu ilk "demosunu" ortaya çıkardı. Bugüne kadar iki intro çıkardıktan sonra Joker'in artık demo efektlerini gözüne kestirmesi, Norveç'ten Turtle'ın Ret'e katılması, Aegis ve Slowhand'in de grafik ve müzik desteği vermesiyle Rising için gerekli şartlar tamamlanmış. Skate'in de linking desteği ile 2010 yılında Türkiye'de kısa süre önce kurulmuş olan bir Commodore 64 grubu böylelikle ilk demosunu çıkarmış oldu.

Şekil 17.

Demo 4 effekt ekranı ve pek çok güzellik barındırıyor. En başta Basic ekranından demoya geçişteki ekranın karakter karakter dolup return logosu kalması çok şık olmuş ve demoyu hemen birkaç sınıf yukarı taşıyor. Plazma zoomer, yüzlerce defa yapılmış olan plazma efektine güzel bir özgünlük kazandırmış. Daha ilerideki demolarda 8x8'den daha küçük pikseller kullanan versiyonları da gelebilir. Sinus noktaları efekti çok temiz ve yeterince hızlı (50 fps'den biraz yavaş). Benim favori bölümüm ise son bölüm. Buradaki logo, üzerindeki renk geçişli çizgi ve asıl effekt olan 3 seviyeli starfield arkasında upscroll gerçekten tam anlamıyla çok şık. Bu partla ilgili çok küçük bir yorumum var sadece o da alttaki logonun ortalanmamış olması. Ama starfield efekti ve efektte yıldızların farklı renklerde olmaları vs. Bence çok çok güzel görünüyor.

Demodaki logolar, charsetler hep gayet profesyonel. Demonun akış hızı çok güzel. En basit bölüm olan greets bölümü bile gayet temiz ve özenli.

Son olarak demonun müziğine bayıldığımı söylemeliyim. Bas ve bateri kanalı, akor sesleri çok dolgun çok güzel. Öndeki filtreli lead ses ve melodiler çok çok ustaca. Hele o birkaç yerdeki işveli glide'lar yok mu. İnsanın içini ısıtan bir müzik. Bende böyle kalkıp dansetme arzusu bile oluşturdu.

Sonuç olarak aslanlar gibi bir demo. Artık bir giriş demosunun gerçekten ötesinde, gururla ayakta durabilecek bir orta-direk demo. Yani C64 scene'in aslında bana göre en çok ihtiyaç duyduğu demo türü.

Çok az farkla compoda NKEOKK'yı geride bırakan Rising yarışmayı kazansaydı da hiç şaşırmazdım. Return ekibine sonsuz saygılar ve önünüzdeki demolarda başarılar.

Hydrogen: Rising, Aegis ve Joker tarafından bir sene önce kurulan Return grubunun ilk demosu. Main coder Joker'ın kendisini c64 alanında hızla geliştirmesi, Return grubunun gün geçtikçe daha iyi ürünlerle karşımıza çıkmasındaki en önemli faktör. Yakın zamanda, Norveç'ten gruba katılan programcı Turtle ve tecrubeli Türk müzisyen Slowhand ile Rising'i yapacak ekip tamamlanmıştı.

Ayrıca, oldukça önemli bir haber de, 7dx 2010 sonrasında tecrubeli artist ve programcı Spritus/Resident'ın (PC), grubun c64 bölümüne, Ragnor/Clash'in de yeni kurulan PC bölümüne katılması.

Return'le ilgili bu bilgileri verdikten sonra, Rising'in incelemesine geçelim.

Demo, klasik c64 ekranından, şık bir geçiş ile ansi(karakterlerden oluşan) Return logosuna bağlanıyor. Karakter silerek ekranda logo bırakma, c64'ün klasik geçişlerinden ancak Return ekranda ansi logo belirdikten sonra, loguyu aşağıya doğru kaydırarak kendi stilini eklemiş ve açılışı zenginleştirmiş.

Bir sonraki bölümde ekranın siyaha fade etmesi ile birlikte, müzik başlıyor ve hi-res Rising logosu ekrana geliyor. Ardından bu logo ekranı terk ediyor. Ve gene başka bir Return logosu altında, Plazma efekti ekrana geliyor.

Bu Return'un yaptığı ilk çok renkli full screen efekt. Klasik 8x8 plazma. Çok zor bir efekt olmasa da Return, efekti olduğu gibi kullanmamış ve challenge içeren bir takım değişiklikler yapmış. Mesela kullanılan palet, standart c64'ün 16 renkli paleti değil. Dolu-boş pixel tekniği ile basit bir renk karışımı yapılarak ara renkler oluşturulmuş ve bu daha yumuşak bir renk geçişi yaratmış. Ardından bir süre plazmayı bu şekilde izledikten sonra, Plazma zoom yapmaya başlıyor. Bu da efekte dinamizm katan, efekti farklılaştıran hoş bir detay.

Bir sonraki efekt gene klasiklerden hoş bir şekilde uygulanmış sinus dot efekti. Sonrasında greetings partı ekrana geliyor. Bu part güzel bir dizayna sahip olsa da, bütün grupların isimlerinin tek tek silinmesini beklemek, demonun hızını olması gerekenden fazla yavaşlatmış diyebiliriz.

Bunun ardından son parta geçiyoruz. Burada bir starfield efektinin arkasında, farklı boyutlarda karakterlerden oluşan creditsler yukarı doğru kayıyor, en sonunda demonun ismi ekranda kalıyor ve demo sona eriyor. Karakter seti ve sprite multiplexerın başarılı bir şekilde kullanıldığı bu efektte, starfield oldukça yoğun ve hoş görünüyor.

Demo grafik olarak beklentiyi karşılasa da, logolarda siyah background yerine biraz daha renkli-eğlenceli patternler kullanılabilirdi. Son ekrandaki logo, hem kullanılan renkler, hem de şekil-şemal itibari ile bence demodaki en başarılı grafik. Başlangıç ve bitişteki Rising logoları ise, demonun ismini, izleyiciye birkaç yerde hatırlatması bakımından isabetli olmuş.

Demonun müziği oldukça güzel bir melodi üzerine kurulu, demoya karakter kazandıran ve demonun kalitesini arttıran bir müzik. Bir önceki introlarında "Last Ninja 3" müziği vs. gibi klasikleri kullanmak zorunda kalan grup için Slowhand'in oldukça büyük bir transfer olduğu aşikar. Son olarak Rising, 7dx 2010'un bence en başarılı ürünlerinden biri idi. Return'un bir sonraki demosunda, yeni transferlerinin de gücü ile çitayı daha da ileri taşıyacağına inancım sağlam.

Spritus: Rising, yarışmadaki tek oldschool platform demosuydu ve açıkçası benim için partideki en büyük sürprizdi. Çünkü Return grubu yeni kurulmuş bir grup, demonun coder'larından Joker yeni bir c64 coderi idi. Daha önceden Joker'in yayınladığı introlarla gelişimini takip ediyorduk. Ben Rising'i perdede gördüğümde yine tek screen'lik bir intro olduğunu zannetmiştim. Ama Return beni yanılttı ve multipart bir demo ile karşımıza çıktı. Üstelik başlangıç için gayet temiz ve güzel bir demo ile.

Demo, c64'ün klasik mavi lacivert ekranında şık bir şekilde beliren ve aşağı doğru kayıp giden bir Return logosu ile başlıyor. Daha sonra siyah bir ekranda dev Rising logosuyla sevimli bir müzik çalıyor. Müzik, uzun bir aradan sonra tekrar scene'e dönen Slowhand'in elinden çıkmış ve çok da iyi etmiş. Çünkü bu şirin demoya çok uymuş bu parça. Fadeout efekti ile kaybolan Rising logosundan sonra plazma partı başlıyor. Plazma efektinin üstünde de hoş bir Return logosu yer alıyor. Bi süre ilerledikten sonra plazma üzerinde zoom-in ve zoom-out yapıldığını görüyoruz. Plazma partının ardından klasik ve eski bir demo efekti olan sinüs dots partı başlıyor. Bu partta ekranın biraz boş kaldığı gözden kaçmıyor. Bir süre sonra güzel bir geçiş efektiyle ekranın alt ve üstünde gri renkli bantlar bouncing efekti ile beliriyor ve diğer gruplara selamlar gönderiliyor. Grup isimlerinin arkasında gezinen boblar da güzel düşünülmüş. Bu part da biterken

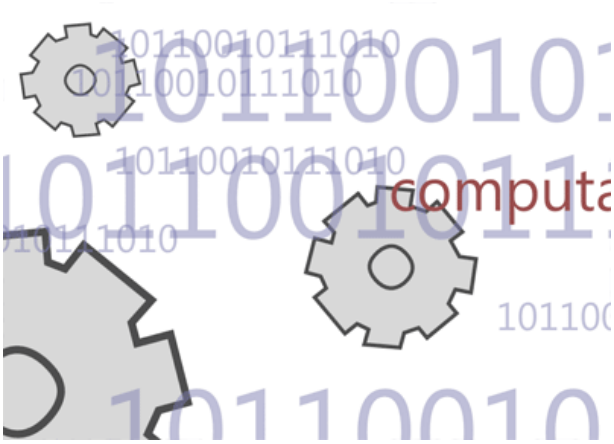
gri bantlar yine güzel bir geçişle son partı haber ediyor. Yine hoş ve renkli bir logo ve onun üstünde starfield efektiyle yukarı kayan kredileri okuyarak demomuz tamamlanıyor. Return grubuna bu güzel demo için teşekkür edelim ve her zamanki temennimizi ekleyelim : Daha iyilerini beliyoruz ;)

Patterns of Isolation

Hydrogen: Bu demoyu bir çok kez izledim. Genel olarak, öncelikle müzik, ardından kullanılan renkler ve hareketlerin yumuşaklığı, kısaca demodaki her şey oldukça rahatlatıcı hatta transa geçirici.

Bu demoyu aslında şöyle görüyorum. Nightlord, çok güzel bir müzik yapmış. Bu müziği öyle kuru kuru sunmak olmaz demiş ve bizlere küçük bir demo sunumu hazırlamış. İyi de olmuş neden mi?

Zira müzik gerçekten çok güzel. demo release olmadan önce de müziği dinlemiştim. Ama demodan sonra dinlediğimde açıkçası müziği daha net algıladım. O kadar hoş, tatlı, naif melodiler ki.



Şekil 18.

Küçük minimalist efektler, müziği çok öne çıkarmış. Senkronizasyonun ustaca kullanımı sayesinde her efektin ekrana gelişi ile birlikte, müzik sizi başka deryalara götürüyor.

İlk beat ile beraber, ana melodiyi dinliyoruz. Ve ilk efektimiz dönen çarklar, 0'lar ve 1'ler ekrana geliyor. 0 ve 1'ler sık kullanılan temalar olsa da, burada bir harekete geçme var. Her şey hesaplanıyor, bizler için hazırlanıyor ve biz bunları, ekrana gelen computing metni ile pekiştiriyoruz.

Klarinet ile beraber gelen simetrik kar tanesi efekti ise, gerçekten klarinet soundunun hoşluğu ile harika birleşiyor. Ardından gelen flüt bölümü ile ekrana basılan boblar, hareketlerindeki yumuşaklık ve görünüşlerindeki şeffaflık ile bizi bulutların üzerinde dolaştırmayı hedefliyor.

Flutün ikinci tekrarında greetsle başlayan harmonik zenginleşme bence demonun en etkileyici kısmı. Burada müzik gerçekten çok duygusal ve insanda çok hoş hisler yaratıyor. O esnada ekrana giren greetsler, etkileyici bir hava yaratıyor.

Son ekran ise, hoş dizayn edilmiş bir final partı ve müzik fadeout oluyor. Demo bitiyor.

Dediğim gibi bu demo harika müziğin etkisinden faydalanan, gayet güzel senkronize edilmiş, karmaşadan uzak basit görsel efektlerle desteklenen hoş bir ürün. Efektler çok büyük zorluklar içermeyen, tamamen görsel tasarımlar ki, Nightlord bu demodaki efektleri muhtemelen C64'de de codelayabilir:)

Burada Nightlord'un hem müzik, hem code hem de görsellik konusunda yeteneklerini kullanarak, aslında kendini çok da yormadan iyi bir katılım gösterdiğini görüyoruz. Dağları devirmeden, bu kadar kısa sürede, kısa ve etkileyici bir demo yapmak da, uzun süredir scenede olmanın getirilerinden biri bence. Bu demoyu, demo yapmaya yeni başlayacan arkadaşların iyi incelemesini tavsiye ederim.

Spritus: Demo tek kişilik dev grup Aesrude'dan Grubun tek elemanı olan Nightlord, adeta "hiç kimseye ihtiyacım yok, ben tek başıma her şeyi hallederim" diyerek kod, grafik ve müziği harmanlayıp güzel bir demo yapmış. Demonun, 7dx 2009'da yayınlanan ve yine birinci olan Patterns of Madness'ın devamı niteliğinde olduğunu düşünüyoruz. Pattern of Isolation tıpkı ilki gibi atmosferik ve yumuşak bir demo. Microsoft'un Directx 2D ekibinde bulunan Nightlord, sanırım bu çalışmalarıyla bir taraftan da geliştirdikleri API'nin yeteneklerini sergilemek amacında. Bu yüzden Windows'un XP'den daha sonraki sürümleriyle çalışabilen Directx 11 gereksinimi var. Nightlord demoyu, parti devam ederken 2 saat içinde (müzik hariç) hazırlayıp göndermiş. Bu bakımdan kendisine saygı beslememek elde değil. Demo kısa olmasına rağmen bi hayli derli toplu ve oluşturmak istenilen havayı vermiş gibi görünüyor. Başlangıçta ekranda beliren çarklar, binary sayılar teknik işlemlerin sanatla harmanlanması mesajını veriyor sanırım. Daha sonra yumuşak bir geçişle, yine beyaz ekran üzerinde, kar kristalleri oluşturan hoş bir fraktal efekti ile devam ediyor. Bu arada Patterns of Madness'takine benzer text mesajlarını da ekranın değişik yerlerine serpiştirilmiş olarak görüyoruz. Demo daha sonra ekranda sinüs bob efektinin daha modern bir varyasyonunun eşliğinde diğer gruplara selamlar göndererek son parta geçiyor. Son part kredilerin gösterildiği basit bir screen'den ibaret olsa da genel uyumu bozmuyor ve demo temiz bir şekilde sona eriyor. Her ne kadar minimal 2D ağırlıklı bir demo olsa da yarışmanın sonucundan da anlayacağınız gibi Nightlord izleyicilerin gönlünü fethetmiş

Genel Demo Yarışması Yorumları

Hydrogen: Evet. 7DX 2010'da yayınlanan dört stuffı da izledik. Gerçekten de Türkiye'de demoscene'de bir ilerleme mevcut. Tabii Breakpoint 2010'da 12nci olan Beacon isimli demo, 7dx 2010'da pek muhtemel olarak bir birincilik alabilirdi. Ancak Beacon içerisinde de, 7d9 birincisi Patterns of Madness'da yer alan conformity partının bir benzerinin mevcut (Belki sadece rastlantıdır, belki ilham başka yerdendir bunu bilemem) olması, doğru yolda olduğumuzun bir işareti olabilir. Breakpoint örneği, hem kat etmemiz gereken daha mesafemiz olduğunu belirtirken, hem de ürettiğimiz materyallerin, uluslarası alanda da ilham kaynağı olabileceğinin altını çizerek heyecan yaratıyor.

Yalnız Kovboylar

Nightlord: Son olarak katıldıkları compolarda tek kalan iki üründen bahsedeceğim.

Rasters (256 byte)

Nightlord: Malesef bu ürün hakkında yorum yapabilecek son adam benim belki de. Ne 256 bayt size coding olayından anlarım, ne de Speccy'den. Buna bir ara çaksa çaksa Ref bir review çakar.

Öz Desert Dream (Wild)

Nightlord: Eveeet geldik partinin en çok iz bırakan ürününe. Bilmeyenler için Desert Dream/Kefrens 90 başlarında Amiga'da çıkmış ve o dönemde efsane olmuş trackmo demolardan biriydi. Daha sonra 2007'de Chorus ve Resource C64 üzerinde de yapmıştı. Yani diyebiliriz ki demo tarihinin önemli kilometre taşlarından biridir. Hatta öyle ki hayatınızda toplam beş Amiga demosu görmüşseniz bunlardan biri kesin Desert Dream'dir (bir diğeri de Hardwired'dır... süperdir... bir de... neyse dağılmayalım)

İşte damarlarında Amiga kanı akan (valla kessen boing ball şeklinde kan damlaları düşüyo adamlardan) Zomco üyeleri, günlük hayatlarında video işleriyle ileri boyutta haşır neşir olsalar da beyinlerindeki piramitlerden ve uzay gemilerinden kurtulamamışlar. Ve sonunda Öz Desert Dream doğmuş.

Bir demoyu gerçek dünyada yapma fikri ilk değil, daha öncede çeşitli varyasyonlarını gördük. Burada öz desert dream'i özel kılan şey o fikir değil. O fikrin uygulanışı. Daha da doğrusu desert dream demosunun çeşitli efektlerini gerçek dünyada hangi objelerle gerçekleştirdikleri. İşte ürünün asıl dehası burada ortaya çıkıyor.

Öncelikle şu anda hala bu demoyu seyretmediyseniz bunu okumayı bırakıp hemen youtube'e gidin. Önce orjinal Desert Dream'i ardından Öz Desert Dream'i izleyin. İzlediniz mi? ... Hayır mı? Hadi kardeşim gidin izleyin. Sonra gelin okuyun yoksa bir sürü kahkahadan olacaksınız.

Eveet şimdi izlediğinize göre ben spoiler korkusu olmadan review'a devam edebilirim. Kağıt ucak, piramitler ve kamera hareketleri... Sünger... Bunlar hep çok iyi fikirler. Aradaki iğne batırılmış tenis topu, sonlardaki stres teli miydi neydi (şu spiral şeyden bahsediyorum) ve hele de masa örtüsü. Parti esnasında herkesin en çok güldüğü anlar oldu sanırım.

Amaaaaa hepsinden de daha çarpıcı bir an vardı ki... Tabii ki kuzu kelleden bahsediyorum. Demonun en epik anıydı gerçekten. O kuzu kellenin arkasında bayağı da emek varmış bu arada.

Emek demişken, bu ürünün arkasında tabii ki baya bir emek ve eminim ki daha da fazla komik hikayeler var. Bunları yakın bir zaman sonra çok daha detaylı okuyabileceğiz. Eski bir dostun geri dönmesiyle...

Zomco, geçen iki sene adım adım partiye katılımını artırmıştı. Bu yıl partiyi sallayan grup oldular. Bakalım seneye neler göreceğiz Zomcodan. Son olarak ünlü bir scener'ın da dediği gibi: ZOMCOOOOOOOOO....

Hydrogen: 7dx2010'un en eğlenceli ürünü, kesinlikle Öz Desert Dream idi. Bir Amiga klasiği , Kefrens'in 1992 tarihli Desert Dream isimli demosunun, wild yorumu olan bu üründe, gerçek hayattan bazı kamera çekimleri ve CG programlarının yardımları ile, demo hemen hemen birebir şekilde yeniden yapılmış. Daha önce de gerçek kamera çekimlerinden oluşan, kaliteli wildlar olduğunu belirtelim. Desert Dream gibi uzun bir demo için böyle bir yükün altına girmek ise oldukça zahmetli bir iş.

Her neyse, ürünün en güzel anlarına değilenim şimdi de:

Öncelikle açılıştaki kartondan yapılmış uzay gemileri, piramitler ve kibritten silahlar ile, intro çok başarılı ve eğlenceli bir şekilde canlandırılmış. Uzay gemilerinin iple sallandırılmaları ve sarı fon kağıdı katlanılarak yapılan piramitlerin koltuk üzerinde durması, iple ilerleyip piramide çarpan karpuz vs. bol bol kahkaha yaratıyor.

Gerçek kurban başı üzerindeki timsah oyuncağı, orjinal desert dreamdeki yaratık resmi ile uyum sağlamış.

Ardından ne olduğunu anlamadığım, esnek bir küp şeklindeki materyal üzerine elle yazılmış Desert Dream logosu ve elle çevrilerek elde edilen X-rotation bölümü, gerçekten kusursuz ve bence demonun en güzel kısmı.

Dolanmış iplik veya sicimlerle yapılan, 11999 dot rekoru da gene harika olmuş.

Misina ile döndürülen tenis topu üzerine iğnelerin bağlı olduğu bölüm de gene çok yaratıcı bir düşüncenin ürünü:)

Ardından elle yazılmış scroll gerçekten çok güzel. Burada devreye giren müzik, kulaklarımızın pasını siliyor.

Birbirine yapışmış tenis toplarının sonrasında, stres halkalarından yapılmış bir tünelin içerisinden başarı ile geçiyoruz.

Dalgalanan pitikareli çarşaf ile yapılan, chessboard efekti ise, gerçekten demonun en önemli bölümlerinden.

Gelelim bu projenin eksiklerine. Öncelikle ilk bölümde müziğin olmaması bence büyük bir handikap. Ayrıca, yukarıda saydığım efektlerin aksine, bazı CG programları ile yapılan efektler, biraz ucuz duruyor. Tamamını gerçek yaşamdan yapmanın çok zor olacağını tahmin ediyorum ancak gördüğümüz harika örnekler, bana Zomco'nun bunu da yapabileceğini düşündürdü.

Sonuç olarak Öz Desert Dream, 7dx'in en eğlenceli ürünlerinden biri olarak tarihte yerini aldı. Artık Zomco'dan beklentilerin daha da arttığını belirtmemize gerek yok herhalde:)

Plazma Künye

Plazma

Amatör Bilgisayar Kültürü

http://www.plazma-dergi.org

Sayi 8 : Eylül 2011 (gecikmiş çıkış tarihi Ocak 2012)

Ekip

Yazarlar:

- Alp Yener (Domino/Zomco)
- Bilgem Çakır (Nightlord/Glance^Aesrude)
- Can Ulaş (Caisson/Ascraeus)
- Emir Diril (Drey/Demodojo)
- Gökhan Sönmez (LW3D/Bronx)
- Kürşad Karamahmutoğlu (Hydrogen/Glance)
- Metehan Alter (Spritus/Resident)

Kapak:

- Kapak Grafiği: Siyam (7dx 2010) - Caner Uyanık
- Kapak Birleştirme: Bilgem Çakır (Nightlord/Glance^Aesrude)

Yardımcı Editörler:

- Emir Akaydın (Skate/Plush)
- 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.

editor (at) plazma-dergi (nokta) org