DAVID MALAN: Tamam. Oyun Geliştirmeye Giriş. Benim adım David Malan ve bu da Colton Ogden. Ve bu sadece CS50 gibi bir sınıfı kabul eden bir sınıftır, bu da kolejler ve yayım okullarının bilgisayar bilimine girişidir, ancak daha genel olarak, herhangi bir dilde önceden programlama deneyimine sahip olduğunuzu ve bu nedenle programlamanın temel yapıları. Ancak Lua veya Lab 2D'de veya sınıfta kullanacağımız çerçevelerin hiçbirinde arka plan olmadığını varsayıyoruz. Bütün bunlar önde. Eğer benim gibiyseniz, muhtemelen bir tür video oyunu ile büyüdünüz. Ve belki de programlamaya başladığınızda, programlama ortamları belki de çok metin tabanlı, siyah beyaz terminal penceresi ve benzeriydi. Ve belki sıfırdan veya Alice veya ötesinde bir dille grafiksel bir şey yaptınız ya da web dünyasındaysanız, bir çeşit daha grafiksel uygulamalar yaptınız, ancak hala oldukça statik. İçeriğin türü ekrana gelir, ardından içerik değişir vb. Ve hepinizin büyüdüğü yederyear'dan bu oyunlardan bazılarını nasıl yaratmaya başladığınızı programlama konusunda biraz daha yeniyseniz, çok daha fazla animasyonun olduğu, çok daha fazla asenkroniklik, bir çok şey oluyor aynı zamanda. Bir çok olay gerçekleşiyor ve hepiniz DEĞİL - sadece bu etkileşimi yakalamak istemiyor, aynı zamanda gerçekleşen olaylara, özellikle de yanınızda veya çevrimiçi başka oyuncularınız varsa, yanıt vermek istiyorsunuz. Ve böylece parkurun yapılandırılacağı yol, birçoğu kendiniz oynamış olabileceğiniz bu çeşitli oyunların bir anlatımıdır. Ancak, dönem boyunca, bu oyunların her birinin bağlamına dalıyoruz ve temel prensiplerden bazılarına, inşa edildikleri yapılara bakıyoruz ve bunları gerçekten bu çeşitli konular hakkında konuşmak için bir çıkış noktası olarak kullanıyoruz. kendi oyunlarınıza entegre edebileceğiniz yetenekler. Ve sonra dönemi noktalamak, nihayetinde bir dizi kilometre taşı olacaktır. Bazıları, daha yakın zamandaki bazı materyalleri pekiştirmek ve daha derin ve projeler üzerinde daha fazla eliniz olduğunda sizi başarıya hazırlayan daha küçük ödevler biçimindedir. , Çünkü gerçekten, proje, kendi oyunlarınızdan bazılarını inşa edeceğiniz veya genişleteceğiniz yerdir. Ve sonra sınıfın kendisi, yarının sonunda kendi final projenizle, bir şekilde veya başka bir şekilde dersin derslerini çeken bir oyun önerme, tasarlama ve uygulama fırsatı ile sona erecek. Böylece, sadece birkaç ay sonra buradan çıktığınızda, sadece oyunlardan adil payınızı değil, aslında birkaç tanenizi de oluşturdunuz. Daha fazla uzatmadan, bir bakış ve yesteryear'ın Pong'unda dolaşmak için işleri Colton'a çevirmeme izin ver. COLTON OGDEN: Teşekkürler David. Size bu kursu öğretmeye başlamak için çok heyecanlıyım çünkü oyun geliştirme aslında beni programlamaya sokan şeydi. 2006 ya da 2007'de bu kitabı burada satın aldığımı hatırlıyorum, 3D oyun programlamasına bir bakış olan 3D Oyun Programlama Hepsi Bir Arada. Ve 2000'lerin sonlarında Torque adı verilen popüler bir oyun motoru bağlamında yekpare bir metindi. Bugünlerde moda değil, ama o zaman oldukça popülerdi ve TorqueScript adlı bir dil kullandı. Ve bu kitabı okuduğumu ve tüm bu kodu gördüğümü hatırlıyorum ve daha önce hiç kaynak kodu gibi görmemiştim ya da hiç programlama ile tanışmamıştım. Ve açıkçası, oldukça korkutucu buldum, çünkü anlamadığım tüm sözdizimine bakıyordum ve oyun geliştirme hakkında hiçbir şey bilmiyordum. Her zaman büyüyen oyunlar oynamış ve büyülenmiştim, ama bilgisayarlarla daha rahat olmaya başladıkça ve daha fazla merak etmeye başladım ve bunun büyük bir meslek olduğunu fark ettim, biraz daha derine dalmaya başladım. Bu benim ilk baskınımdı. Ve oldukça gizli olan bir TorqueScript için kaynak kodunu inceledikten sonra biraz zaman geçirdikten sonra, yüzde yüz simge ve dolar işareti, aklıma gelmediğim tuhaf şeyler. Geri döndüm, gerçekten programlamanın temellerini ve Python ve c ve c ++ gibi diğer dilleri öğrenmeye başladım ve programlama ve bilgisayar bilimini çok sevdim. Ve işte Torque'un o zaman nasıl göründüğünün bir görüntüsü. Gerçekten de bugünlerde Birliğin öncüsü gibiydi. Her ne kadar, bence, Birlik işleri çok daha iyi yapıyor. Daha erişilebilirdi, daha fazla moda olan ve popüler olan ve zaten başka alanlarda bulunan diğer insanlar tarafından kullanılan dilleri kullanıyor. Ve böylece kursun arka ucunda Birliği ele alacağız. Yani biz ' ağırlıklı olarak 2D oyun geliştirmeyi kapsayacak. Ancak bugün Pong bağlamında başlarken ele alacağımız konular bu mermi noktaları. Lua, kurs boyunca ağırlıklı olarak kullanacağımız dil olacak, bu da Python ve JavaScript'e çok benzeyen dinamik bir betik dilidir. Lua ile çizim, ses, giriş, vb. İçin tüm yöntemlerini ortaya koyan bir çalışma zamanı ve bir çerçeve olan LOVE 2D'yi birincil kod olarak ele alacağız, böylece kodu çok hızlı yazmak çok kolay, ancak çok iyi sonuçlar elde edin. Ve onların çerçevesi için belgeler bence mükemmel. Bugün oyun geliştirme ile ayaklarımızı ıslattığımız için birkaç temel prensipten bahsedeceğiz. Şekil çizme, metin çizme gibi şeyler, bunların ikisi de Pong'un çok büyük yönleri, Ekranda hareket eden şekiller ve metinlere dayanan çok basit bir oyundur. Delta Time ve Velocity hakkında konuşacağız, ki Delta Time muhtemelen herhangi bir oyun çerçevesinde veya motorda takip ettiğimiz en önemli değişkenlerden biridir, bu sadece yürütmenin son çerçevesinden bu yana geçen süredir. Oyunumuzda, AŞK 2D cinsinden saniye, saniye kesir cinsinden ölçülür. Oyun durumu hakkında konuşacağız, çünkü oyununuzda bir durum olabilir. Başlık ekranında olabilirsiniz, oynuyor olabilirsiniz, bir menüde olabilirsiniz. Açıkçası bu çok önemli olacaktır çünkü içinde bulunduğunuz duruma bağlı olarak farklı güncelleme mantığı ve oluşturma mantığı istiyorsunuz. C'den tanıdık gelmeyenler için temel nesne yönelimli programlama hakkında konuşacağız. Temel olarak, verilerimizle, oyun nesnelerimizden herhangi birini, kendileriyle ilgili değişkenlerin, bu veriler üzerinde çalışacak işlevlerle bir araya getirileceği şekilde kapsüllemenin bir yolu. Bu nedenle, kodunuzda izlemeniz gereken tüm bu farklı nesneler için 20 farklı değişkene sahip olmak yerine, her bir nesne, konumu veya onunla ilgili başka bir şey gibi tüm kendi bilgilerini izleyebilir. Bugün hit kutulardan, ağırlıklı olarak, kutu çarpışması bağlamında konuşacağız, çünkü sadece kürekler ve bir top olan Pong hakkında konuşacağız. Bunlar dikdörtgenler. Ve eksen hizalı bağlı olarak adlandırılanlarla çarpışacaklar - eksen hizalı sınırlayıcı kutular, bu da iki kutunun çarpışıp çarpmadığını, döndürülmüş isabet kutularını hesaplamanın aksine, bu biraz daha karmaşık. Ve son olarak, ses efektleriyle parlatacağız, çünkü bence bu cilalı katmanı eklemek önemlidir ve hepsini birbirine bağlar ve daha uyumlu bir bütün gibi hissettirir. Bu yüzden, bir anda deponun bağlantısını göstereceğim örnekler ile birlikte izlememiz gereken iki önemli şey LOVE 2D'yi yüklüyor. Bu çok basit bir süreç. Buradaki ilk bağlantı sadece bir indirme bağlantısıdır. Bu nedenle, tüm büyük işletim sistemleri için kullanılabilir. Linux, Mac ve Windows. Ve sonra aşağıdaki Başlarken bağlantısı size nasıl başlayacağınıza dair bazı ipuçları verecek, aslında Mac'inizde makinenizde çalıştığı uygulamada gerçek çalışma zamanı yürütülebilir iAlias. Bu yüzden bash profillerimde kolayca ana nokta Lua dosyası olan herhangi bir dizine aşk alanı noktasını yazabilirim ve çok basit bir yerde çalıştırabilirim. Diğer işletim sistemleri için de sayfada benzer talimatlar bulunmaktadır. Ve bu, bugün kullanacağımız tüm kaynak koduna sahip olan repo. Ve bunu takip edebilmeniz için 13 farklı alt pozisyondan oluşan bir dizi halinde yapılandırdım ve sıfırdan başlayarak tam olarak uygulanmış bir oyuna kadar Pong'u geliştirebiliriz. İlk konuşacağımız şey Lua'nın ne olduğudur. Kursun yaklaşık% 75'inde Lua kullanacağız. Çok popüler bir dinamik betik dilidir. Ay için Portekizce, ve 90'ların başında, bu kod tabanlarına kod ekleme ve bunları yeniden derleme konusunda zaman kazanmak için derlenmiş kod tabanları için bir yapılandırma dili ve çalışma zamanı dili olarak icat edildi. Özellikle bilgisayarların daha yavaş olduğu 90'ların bağlamında, uygulamanızın temel işlevselliğini Lua'ya göstermek için çok daha hızlı ve çok daha kolay, böylece dinamik olarak çalıştırabilir ve daha sonra derlenmiş kodunuzla anında etkileşim kurabilirsiniz, sadece yeni bir davranış elde etmek için dakikalar, potansiyel saatler, yeniden derlemek ve beklemek zorunda değilsiniz. Bir masa kavramına odaklanan bir dildir. Lua'daki temel değişkenlerin yanı sıra hemen hemen her şey tablolardır. Tablo aslında Python'daki bir sözlük veya JavaScript'teki bir nesnedir. Çok benzer. Niyet, daha büyük uygulamalarda gömülü kullanım için, ve bu büyük uygulamalar bağlamında kullanılması amaçlanan Lua'nın doğası, oyun motorlarıyla etkileşim için mükemmel olduğu anlamına geliyordu. Çünkü oyun motorları, hız amaçları için geleneksel olarak kodlanmış kod tabanlarına mükemmel bir örnektir. Ancak küçük işlevsellik eklemek ve daha sonra yeniden derlemek ve potansiyel olarak tüm stüdyonuzun saatler almasını sağlamak çok hantal olabilir. Bu yüzden hızlı bir şekilde gelişmemizi sağlamak için Lua ve derlenmiş bir oyun çerçevesi olan LOVE 2D'yi kullanacağız. JavaScript ve Python'a benzer. JavaScript için biraz daha fazla. Ve çok mükemmel, çünkü başlangıçta bir yapılandırma dili ve sadece bir çeşit tutkal tabakası olarak tasarlandı. Veri ve kodu bir arada depolamak için çok iyi, neredeyse aynı. Yani AŞK 2D hızlı bir 2D oyun geliştirme-- geliştirme çerçevesi. O' C ++ ile derlenir ve çok verimli çalışır. Çünkü Lua'da çalıştırmamıza rağmen ve 2D oyun geliştirme bağlamında ihtiyacınız olan her şey için modüller olarak çok basit. Sadece 2D oyun geliştirme resmi olarak, tanıdığım bazı insanlar küçük küçük 3D deneyler üzerinde çalışıyor, ancak henüz resmi bir şey yok. Ancak grafik, klavye girişi, matematik, temelde 2D oyun geliştirme bağlamında isteyebileceğiniz her şeye sahiptir. Tamamen ücretsiz. Taşınabilir. Hatta mobil cihazlarda ve web'de de çalıştırabilirsiniz. Ve prototipleme için mükemmeldir, bir oyunu LOVE 2D'de yayınlamak zorunda olmasanız bile, LOVE 2D'de bir şeyi kırmak ve sonra bunu hangi çerçeveye veya motora taşıyacağınız harika ve kolay ve hızlıdır. gerçek dünyada kullanmak. Bazı gerçek somut kodlara bakmadan önce, göz önünde bulundurmamız gereken en temel şey oyun döngüsünün ne olduğudur. Yani bir oyun, temelde, bir süre doğru ya da bir süre gibi, sadece sonsuz bir döngüdür. Sadece bu durumda, bu döngünün her yinelemesi, arka arkaya tekrar tekrar bir dizi adım yapıyoruz. Girişi işliyoruz, böylece görüyoruz, kullanıcı klavyede bir tuşa bastı, joysticklerine dokundular, fareyi hareket ettirdiler, fareyi tıkladılar. Varsa, bunu güncellememize beslemeliyiz. Bunu takip etmemiz ve oyun durumumuzda bu girdiye dayanan her şeyi değiştirmemiz gerekiyor. Bu yüzden küreklerimizi hareket ettirmeliyiz, çarpışmayı tespit etmeliyiz, tüm bunları kaydetmeliyiz ve sonra ne güncellenirse, bunu yeniden söylemek istiyoruz. Bunu yapmak istiyoruz - nerede değiştiğini render edelim ki, ekranımızda görüyoruz - görsel olarak, oyun dünyamızda bir şeylerin gerçekten değiştiğini ve onunla etkileşime girdiğimizi ve dinamik bir şeyle etkileşimde bulunmak. 2D oyunlar bağlamında, dünyaya bakmanın en temel yolu, tıpkı lise, x ve y eksenlerindeki geometride öğrendiğimiz gibi 2D koordinat sistemidir. Bu durumda, tipik olarak öğrendiklerimizden biraz farklıdır. Lisede, xy kökenlerinin, bir tür sol alt, y pozitif yükseldi, negatif aşağı, pozitif x sağa ve negatif x sola gittiğini öğrenme eğilimindeyiz. Ama bu durumda, aslında sol üstten başlıyoruz ve sonra y pozitif aşağı, y negatif yukarı, x pozitif sağ, x negatif sola gidiyor. Ve oyunumuzda çizmek istediğimiz her şeyin ekranda görsel olarak görülmesi için çizmek için bir x ve y koordinatına sahip olması gerekir. Bugünün hedefi, oldukça düşük bir seviyeye başlayacağız ve bugün ve gelecekteki sınıflardaki örneklerle yolumuza devam edeceğiz. İlk oyunumuz tartışmasız en basit, ama aynı zamanda, tüm zamanların en ünlü oyunlarından biri, 1972'de piyasaya sürülen Pong. Ve Pong'un özü, ekranın sol tarafında bir kürek var, ekranın sağ tarafındaki kürek, kim rakibin kürek ekranın kenarına topu alarak 10 puan alırsa, kazanır. Ve böylece bugün dersimizde kapsam, her şeyden önce ekrana şekiller çizmek istiyoruz, çünkü topumuzu ve-- top ve kürekler renderımızı bu şekilde elde ediyoruz. Ve bunlar sadece dikdörtgenler. Bu küreklerin 2B konumunu kontrol etmek istiyoruz, çünkü yukarı ve aşağı hareket etmelerini ve topun da hareket etmesini istiyoruz. Kürekler ve top arasındaki çarpışmayı tespit etmek istiyoruz, çünkü topu küreklerden saptırmak ve tavandan ve yerden saptırmak için bu şekilde elde ediyoruz. Ayrıca, ekranın kenarlarının ötesine geçip geçmediğini nasıl tespit ettiğimiz, böylece bir oyuncu bir puan kazanır. Ve sonra bir tür geri bildirim için ses efektleri eklemek ve kendimizi biraz daha oyuna sokmak istiyoruz. Ve sonra skorlama, çünkü nihayetinde oyunun amacı rakibinizi yenmektir, bu yüzden ilk önce kimin 10 puan topladığını görmenin bir yolunu istersiniz. Ve şimdi repoda bir dizi örneği inceleyeceğiz. Pong Zero'ya bakarsak, bunun Day Zero Update olarak adlandırılacağını ayarladım. O' oyunların x güncellemesi olarak önemli içerik güncellemelerini yayınlaması için birçok oyun arasında bir eğilim. Sadece sevimli olmak için, sanırım burada her bir örneği, Bir Şey Güncellemesi olarak adlandıracağız. Ve bu yüzden dizinin Pong Zero Repo'suna gideceğim, GitHub repo. Ve eğer burada Pong Zero'ya bakacak olursak, burada Gün Zero Güncellemesi dediğini görebiliriz. Her şeyi oldukça ağır bir şekilde yorumladım, böylece kodu okuyorsanız, neler olup bittiğine dair bir fikir edinebilirsiniz. 23. satırda, sadece bir pencere genişliği ve bir pencere yüksekliği bildirerek başlayacağız. Ve bunlar sadece uygulamamızın geri kalanında erişilebilecek sabit değişkenlerdir. Bu yüzden sadece 1280 x 720'yi keyfi bir çözünürlük olarak ayarlıyorum. Çok önemli değil. Burada bakmamız gereken önemli bir şey, 29. satır, love.load adlı bir işlev kullanıyoruz ve aslında buradaki slaytlara geri döneceğim. Birkaç işleve bakacağız ve bunları ele alacağım ve koda çok fazla detaylı bakmadan önce ne yaptıklarını anlatacağım. Bu yüzden love.load sadece bize LOVE, LOVE 2D tarafından verilen bir işlevdir ve üzerine yazıyoruz. Ona davranış veriyoruz, ne yapacağımızı söylüyoruz. Ve LOVE 2D buna main.lua dosyamızda bakacak. Pong Zero'ya bakıyorsak, sadece bir main.lua dosyası olduğunu göreceksiniz. LOVE 2D sadece bir main.lua dosyası bekler ve main.lua dosyasını çalıştırır ve bu main.lua dosyasından dizindeki diğer dosyalara başvurabilirsiniz. Etkili olarak bizim bootstrap'ımız. Sevgiyi geçersiz kılacağız. Uygulamamızın başlangıcında yürütmek istediğimiz her şeyi yükler. Bu sadece bir başlangıç ​​işlevidir. Ayrıca, tüm bu davranışı yukarıdaki fonksiyonun dışında da tanımlayabiliriz, ancak love.load içinde bulmak iyi bir uygulamadır, böylece kodunuzu okuyan biri bilir, tamam, bu tüm başlangıç ​​kodunun gerçekleştiği yerdir. Love.update (dt) çok önemli bir işlevdir. Bu işlev (dt) adlı bir değişkeni alır. Aşk onu bir işlevde geçirir. Kendi davranışınızla üzerine yazacaksınız ve Aşk bunu her kareyi yürütecek, delta zamanında geçirecek ve uygulamanızı ne kadar zamana bağlı olarak değiştirmek için bu işlevde delta zamanını (dt) kullanabilirsiniz. geçti. (dt), bilgisayarınızın ne kadar yavaş olduğuna bağlı olarak, her zaman saniyenin bir kısmı olacaktır. Ancak, tipik olarak, saniyenin altıda biri. Ve tüm kare hızlarında eşit davranışlar elde etmek için oyununuzdaki her şeyi bu miktara göre ölçeklendirebilirsiniz. Love.draw, güncelleme ve çizim arasındaki diğer büyük işlevdir. İkisinin ikisi, tartışmasız en önemli işlevlerden. Love.draw, tanımlayacağımız, çizim davranışımızın tamamı, içindeki renderleme davranışımızdır. Ve burada küreklerimizi çizebiliriz, topumuzu çizebiliriz. Ve sonra güncelleme, küreklerin konumunu ve benzerlerini değiştirmeyi sevebileceğimiz yerdir. İlk örnekte iki önemli fonksiyon daha ele alacağız. Love.graphics.printf printf ve C'nin LOVE 2D analogudur. Farkı bu printf aslında bir konsola karşı ekrana fiziksel olarak çizmemizi sağlar. Ona dize olarak bir metin, bir x ve bir y koordinatı ve isteğe bağlı olarak bir genişlik ve bir hizalama veriyoruz ve ' metni xy'de çizer, ancak genişliği de dikkate alır ve hizalamayı da dikkate alır. İle hizalamanın ne kadar olduğu ve hizalama hizalama modudur. Yani eğer x sıfır genişlik, pencere genişliğimiz ve sonra hizalama merkezi deriz, sıfır ile pencere genişliğimiz arasında gidip merkezle hizalar. Böylece bunun merkezimizi metnimizi hizalama etkisi olacaktır. Ama aynı şekilde kolayca söyleyebiliriz, doğru ve bu ikisi arasında doğru hizalayacak ve ekranı oluşturma efektine sahip olacak - metni ekranın sağ kenarı boyunca oluşturma. Ve son olarak, love.window.setmode bir genişlik, yükseklik ve isteğe bağlı bazı parametreler alır. Bu parametreler V senkronizasyonu ve tam ekran gibi şeylerdir ve aslında penceremizi kuracak ve ekranda görüntülenmesini sağlayacaktır. Ve böylece burada kaynak kodumuza geri dönersek, 29. satırda, aşkın üzerine yazıyoruz. Yük. Yukarıda 1280 x 720 olarak tanımladığımızı hatırlayan love.window.setmode, pencere genişliği ve pencere yüksekliğini geçiyoruz. Bir tablodan geçiyoruz. Bu bir tablonun sözdizimidir, bu süslü parantezler. Ve anahtarları ve değerleri tanımlama şeklimiz sadece orada eşittir. Böylece tam ekran yanlış, yeniden boyutlandırılabilir yanlış, V senkronizasyonu doğru olur. Bu yüzden tam ekran olmayacak, yeniden boyutlandırılamayacak, ancak monitörünüzün yenileme hızıyla senkronize edilecek. V senkronizasyonu burada, dikey senkronizasyonun kısaltmasıdır. Ve sonra 40. satırda, love.draw'ın üzerine yazıyoruz ve bunun love.graphics.printf işlevi var, içerdiler ve diyoruz ki, dize geçiyoruz, merhaba, Pong. Biz' x sıfıra başlayarak, y pencere yüksekliğine 2, eksi 6'ya ayarlıyoruz. AŞK 2D'deki varsayılan yazı tipi boyutu 12 piksel yüksekliğinde. Bu yüzden altıda yukarı kaydırıyoruz, böylece ekranda dikey olarak mükemmel bir şekilde ortalanmış. Ve sonra hizalama miktarını, genişliği, pencere genişliğine ayarlayacağız, böylece onu penceremizin tüm genişliği içinde hizalayacak. Ve şimdi orta hizalamaya ayarlıyoruz. Yani x sıfırdan başlayarak tüm penceremizin ortasına hizalanacak. Ve eğer Pong sıfıra gidersek ve aslında onu çalıştırırsak, bunu yapmanın etkisi vardır. Sadece varsayılan yazı tipimizde, varsayılan boyutta, merhaba, Pong'da ekranın tam ortasında oluşturuyoruz. Bu yüzden çok heyecan verici bir örnek değil, ama LOVE 2D'nin en önemli işlevlerini sergiliyor, böylece slig ile başlayabilirizhtly daha ilginç örnekler. Yani, ilk içerik güncellememiz Düşük Çözünürlüklü Güncelleme. Pong'u geliştiriyoruz ve Pong eski bir oyun. Yazı tipinin oldukça yüksek çözünürlüklü olduğu yere baktığımız örnek gibi görünmüyor. Biraz daha retro görünen bir şey istiyoruz. Yapmak istediğimiz şey, çözünürlüğümüzü 1972'de piyasaya sürülen bir oyundan almış gibi göstermek. Bu yüzden burada yapacağımız birkaç önemli işleve bakmak. Yani Pong One'ın bu işlevleri var. Yani love.graphics.setDefaultFilter. Bunun amacı, uygulamamızda her yazı tipimiz veya resmimiz olduğunda, varsayılan olarak bir filtre uygulanacaktır. Bu yüzden varsayılan olarak bilinear bir filtreye gider. Ne olacak, bunun etkisi, temelde, bir dokuyu büyüttüğümüzde veya küçülttüğümüzde, ' Bunu düşünecek - çok sarhoş görünmemek için biraz bulanık olmasını istediğimizi varsayacağız. Hangi belirli bağlamlarda iyidir. Daha yüksek çözünürlüklü 2B oyun geliştirme için bu iyi, ama göreceğimiz gibi, retro oyunlar bağlamında bu özellikle iyi değil. Retro oyunların çok 2D, keskin, pikselli bir estetiği var ve bunu korumak istiyoruz. Ve böylece bu, varsayılan bir filtre belirlememizi sağlar. Kısa bir süre içinde kullanımda olduğunu göreceğiz. Daha önce gördüğümüz oyun döngümüzün giriş aşaması olan bir başka önemli - çok önemli işlev, love.keypressed (key), oyunumuzun bu yönü ile etkileşime girmemize izin verecek olan şeydir. Love.keypressed (key), LOVE'nin main.lua'da beklediği bir geri arama işlevidir. Üzerine yazacağız. Bir anahtardan geçer, ve bir tuşa bastığımızda bu fonksiyon LOVE 2D tarafından her seferinde çağrılır. Basılan bir tuşu algılar ve bu işlevi çağırır. Burada tanımladığımız ne olursa olsun, onu çağıracak ve belirli anahtarları almak ve bu giriş üzerinde belirli işlemleri yapmak için ayarlayabiliriz ve bir dize alır. Demek ki-- kaçış tuşuna basarsak, anahtar o işlevdeki dize kaçışına eşit olur ve buna erişebiliriz. Ve başka bir önemli işlev, aşk.event.quit. Bu, uygulamadan çıkmanın çok basit bir etkisine sahiptir, ancak bunu kendimiz yapmak yerine kodda çağırabiliriz. İşte doku filtrelemenin neye benzediğine bir örnek. Nokta filtreleme, en yakın komşu filtreleme ile aynıdır. Bilineer filtreleme sağda gösterilir, oldukça bulanık görünüyor. LOVE 2D varsayılan olarak hem yazı tiplerine hem de dokulara uygulanır. Ve bunu bir örnekte göreceğiz. Aslında iki farklı tarzda çalıştırabilirim. Bu yüzden repoda Pong One'a giderseniz ve onu çalıştırırsak, burada görüyoruz, merhaba, Pong şimdi havaya uçuyor. Ve aslında neden patladığını görmek için biraz daha koda bakacağız. Ama kodumuza geri dönersek, Pong One'ı açayım. Main.lua'ya gidin, sonra bunu sadece bir saniye içinde açıklayacağım, ama bunu yorumlayayım ve farkı burada görelim. Çok daha bulanık göründüğünü görebilirsiniz. Ve bu gerçekleşen varsayılan doku filtrelemesi. Dediğim gibi, sadece dokular için değil yazı tipleri için de geçerlidir. Ve bu istediğimiz estetik değil. O halde Pong One'a üstten başlayarak detaylı olarak bakalım. 28. satırda bir kütüphane ediniyoruz. LOVE 2D uygulamanızda veya LOVE uygulamanızda bir kütüphane bu şekilde elde edilir. Sadece gerektirir ve kitaplığın adı eşittir. 1280 x 720 penceremizi alıp 432 x 243'te sanal çözünürlük penceresine dönüştürmek için kullanacağımız şey, oyunumuzu daha düşük çözünürlüklü bir his olarak düşünmeye başlayabilir ve 432 x 243 piksel boyutunda, ancak yine de keyfi olarak boyutlandırılmış bir pencerede oluşturuyor. Bu durumda, daha önce gördüğümüz 1280 x 720 penceresini koruyoruz. Love.load fonksiyonumuza giderseniz, bunun 47. satırda kullanıldığını görüyoruz. Love.window.setmode yerine, artık sanal kurulum gerektiren push kütüphaneleri kurulum ekranı fonksiyonu olan push setup ekranını kullanıyoruz, sanal yükseklik, normal pencere genişliğimiz ve pencere yüksekliğimiz ve daha önce olduğu gibi aynı tablo. Ve bu, 1280 x 720 beton boyutlarımıza sahip bir pencere, ancak 432 x 243 sanal çözünürlüğüne sahip bir pencere oluşturma etkisine sahiptir. Ve şimdi, oluşturduğunda, kısa süre içinde göreceğimiz gibi, zaten gördük, aslında, büyütülmüş. Bize daha düşük bir çözünürlük verme etkisi vardır. Ve on line - 58. satırda, eğer love.keypressed fonksiyonuna bakarsak, eğer anahtar dize kaçışına eşitse, o zaman love.event.quit. Şimdi girdi işleme var. AŞK 2D'nin uygulamamızda arayacağı love.keypressed (key) 'i geçersiz kıldık ve gerektiğinde çağırıyoruz. Ve sonra biz sadece oraya bakıyoruz. Anahtar kaçarsa, o zaman love.event.quit. Ve uygulamayı çalıştırırsam, şimdi klavyemde escape tuşuna basıp bırakabilirim ve bir Windows uygulamasında quit komutunu veya X'i tıklatmam gerekmez. Ve 70 satırındaki love.draw işlevinde bir şey daha değiştirdik. Şimdi push kütüphanesini kullanıyoruz. Yapmamız gereken-- bir tür devlet makinesi gibi işlev görüyor, çünkü itme uygula başlat ile sanal bir çözünürlükte render yapmaya başlıyoruz ve sonra uygula sonunu it, ve aradaki her şey, bu aslında nasıl Açık'a çok benziyor Git çalışır. Çok fazla ayrıntıya girmeyeceğiz. Ancak bu, Open Go programlamanın ne kadar işe yaradığına ruhen çok benzer. Uygula başlangıcı it, uygula ucunu it Bunların arasında, her ne diyelim, bu sanal çözünürlükte render edeceğiz. Ve böylece aynı love.graphics.print (f) fonksiyonunu çağırıyoruz, merhaba Pong Zero, sanal yüksek iki eksi altıya bölündü. Aynı parametreler. Ve her şeyi renderleme etkisi var. eski cisim cilt üzerinde hâlâ devam ediyor - doku filtrelemesi bize büyütülmüş metnimizi vermenin bir etkisi olarak. Yani aynı metin, aynı boyut, ama şimdi oluşturma penceremiz çok daha küçük. Yani-- Şimdiye kadar bunun nasıl çalıştığıyla ilgili sorularınız mı var? TAMAM. Muhteşem. Bu yüzden metni doğrudan ekranın içine aldık, ancak henüz Pong'a yakın değiliz, bu yüzden ilk büyük şey, sanırım, bu bizi dikdörtgen güncelleme olarak adlandıracağımız yöne yaklaştıracak . Bu yüzden bakmamız gereken bazı önemli fonksiyonlar. Love.graphics.newFont. Varsayılan yazı tipi, sanırım Arial. Uygulamamızda Arial istemiyoruz, çünkü biraz daha retro görünen bir şey istiyoruz. Daha alakalı görünen bir şey istiyoruz. Love.graphics.newFont temel olarak klasörümüzde bulunan bir font dosyasına giden bir yol alacaktır. Pong 2 klasöründe bir font.ttf dosyası ve bir boyut göreceksiniz. Çünkü başlattığımız her yazı tipi nesnesinin bir boyutu olması gerekir, çünkü yazı tipi nesneleri değişmezdir. Bir kez inşa edildiğinde, değiştirilemezler, böylece boyut bazında boyut olarak tahsis edilmeleri gerekir. Love.graphics.setFont, bu işlev çağrısından edindiğimiz yazı tipi nesnesini alacaktır ve burada ayarlayabiliriz ve LOVE 2D'deki etkin yazı tipini bu yazı tipi olarak ayarlar. Aşk, daha önce olduğu gibi aynı anlamda bir durum makinesidir, herhangi bir zamanda aktif bir yazı tipine sahip olacağı ve aradığınız yazdırma işlevleri ne olursa olsun, o anda etkin olan yazı tipini kullanacaktır. Ve bu aynı zamanda, ekrana ne renk vermek istersiniz, ne olursa olsun geçerlidir. Bir yazı tipiniz varsa ve belki de kırmızıyla işlemek istiyorsanız, LOVE 2D'nin aktif rengini de kırmızıya ayarlamanız gerekir. Love.graphics. clear, bir RGBA dörtlü alan ve ekranı bu renkte yıkayan bir işlevdir. Sadece bu renkte ekranı silmenin basit bir etkisi var. Sadece düz renkli arka planlar çizmek için kullanışlıdır. Ve sonra son işlev, muhtemelen en önemli işlev, love.graphics.rectangle. Ve bu, göreceğimiz ilk işlev, aslında metnin ötesinde bir şey çizerek ekrana geliyor. Dolgu veya çizgi, x ve ay, genişlik ve yükseklik olabilen bir mod alır ve bu modda bir dikdörtgen çizer. Doldurulmuş ya da doldurulmuş bir dikdörtgen ya da bir çizgi dikdörtgeni. Sürecek-- içeri girdiğimiz genişlik ve yükseklikle xy'de çizecek. Öyleyse devam edelim ve bunun gerçekte uygulandığını görebildiğimiz Pong 2'ye bir göz atalım. Yani Pong 2, bizim font.ttf var dahil ettik. Ve sonra bir main.lua. Ve bu arada, son örnekten bahsetmeyi unuttum, ihtiyacımız olan kütüphaneyi aynı dizinde de itiyorum. Ve sadece dosya dizinde olduğu sürece gerektirebilir, sadece yükleyecektir. .Lua belirtmeniz gerekmez. Bazı dize gerektiğinde, bunun .lua son ekiyle devam ettiğini varsayar. Bu yüzden burada anaya bakıyoruz. Dikdörtgen güncelleme olan Pong 2'de lua. Yani 28-34. Satırlarda hepsi aynı. Push alıyoruz. Genişlik ve yüksekliğimiz sanal ve fiziksel olarak elimizde. Love.load fonksiyonumuzda, küçük yazı tipinin bir love.graphics.newFont olduğunu ilan ederek 43. satırdayız, bu da font.ttf yolunu veriyor çünkü orada aynı boyutta sekiz boyutta. Ve bu, daha sonra gerektiğinde etkin yazı tipi olarak ayarlayabileceğimiz bir yazı tipi nesnesi, küçük yazı tipi oluşturacaktır. Aynı dizinde 78. satıra inersek, love.graphics.clear dediğimizi görüyoruz ve bu yüzden onu bir renkte geçiriyoruz. Google Görseller'de Pong'un bazı görüntülerini örnekledim ve beğendiğim bir arka plan gri gördüm, bu yüzden 40, 45, 52, RGB ve sonra 255 tamamen opak, yani saydamlık yok, alfa bileşeni. Ve sonra 81. satırda yaptığımızla aynı baskı işlevini yapıyoruz. 89. satırdan 95. satırın altına, aslında love.graphics.rectangle diyoruz. Ve bunlar iki kürek ve sonra topu çiziyor. Yani, not, love.graphics.rectangle doldurma modu, çünkü küreklerin top gibi tamamen doldurulmasını istiyoruz. Ona 1030 xy ve 520 genişlik yüksekliği veriyoruz. Ve 992'de biz-- ve bu onu sol üst köşeden biraz kaydırılmış olarak çizme etkisine sahip olacak, beş piksel genişliğinde, 20 piksel yüksekliğinde. 92. satırda, aynı şeyi yapıyoruz, ancak sanal genişlik eksi 10'a gidiyoruz. Böylece, ekranımızın sağ kenarına, sanal genişlik eksi 10'a gidiyoruz. Yani 432 eksi 10. Böylece 422. Ve sonra sanal yükseklik eksi 50. Yani biraz olacak - ekranın altından biraz yukarı çıkacak. Bu yüzden sol üst kürek ve sağ alt kürek var. Ve sonra top ölü merkez. Bu yüzden oturuyoruz - başka bir grafik yapıyoruz - dikdörtgen çağrı. Sanal genişlik eksi ikiye bölünür. Ortada, eksi iki, çünkü topumuz iki piksel genişliğinde ve iki piksel yüksekliğinde olacak. Sanal yükseklik ile aynı şey, eksi iki. Sanal yüksekliğimiz ikiye, eksi ikiye bölündü. Bu yüzden devam edip Pong 2 dizinine CD yazıp çalıştıracağım. Bunun ortada yeni yazı tipimiz var. Bu yüzden Arial yazı tipinden daha önce olduğu gibi güzel ve retro görünüyor. Burada beş piksel genişliğinde ve 20 piksel yüksekliğinde bir dikdörtgen var. Ortada, dört piksel genişliğinde ve dört piksel yüksekliğinde bir top ve daha sonra sağ altta, soldaki kürekle aynı boyutlarda olan bir kürek. Yani Pong'a çok benziyor. Hiç etkileşimli değil, ancak uygulamamızın nasıl görünmesini istediğimizi hissettiriyoruz. Biz çoğunlukla taslağı var. Peki bunun nasıl çalıştığıyla ilgili sorularınız mı var? TAMAM. Muhteşem. Yani Pong 3. Şu anda uygulamamızla hiçbir etkileşimimiz yok ve kürekleri hareket ettirmek istiyoruz. Biz sadece bir görüntüye sürekli bakmak istemiyoruz. Yani kürek güncellemesi bu sorunu bizim için çözecek. Aslında ilk çeşitimizi alacağız - uygulamadan çıkmak için kaçışa basmanın ötesinde, onunla dinamik olarak etkileşime girme hissi elde edeceğiz. Bu örnekte bakacağımız önemli fonksiyon love.keyboard.isDownsomekey. Ve bu doğru-- bu bir Boole işlevidir. Bir dize olarak ilettiğimiz tuşun şu anda bu karede basılı olup olmamasına bağlı olarak doğru veya yanlış döndürür. Bu yüzden sadece doğru veya yanlış döndürür. Şimdi devam edip demoya bir göz atalım. Devam edip Pong 3'ü çekeceğiz. Main.lua, içerdiler. Pong 3'e bakarsanız, kürek hızı olarak tanımladığımız yeni bir sabitimiz var. Ve bu sadece iyi bir hız olduğunu bulduğum keyfi bir değer. Ancak kürek çekme hızımız bu kadar hızlı. Bunu delta zamanına göre ölçeklendireceğiz, bu yüzden kaç saniye geçtiğini çoğaltacağız. Tipik olarak, son kareden bu yana bir saniyenin bir kısmı. Bu nedenle, bilgisayarınızın saniyede 10 kare veya saniyede 60 kare hızında çalışmasına bağlı olarak zaman içinde aynı mesafeyi hareket ettirecektir. Burada 63. satıra gidersek, iki yeni değişken de ayarladım. Oyuncu bir puan ve oyuncu iki puan. Her ikisi de sıfır olarak başlatılır. Bu örnekte skorun bir kısmını da ekleyeceğiz. Burada 49 satırına dikkat edin, değişmez nesneler oldukları için yazı tiplerini boyutlarına göre nasıl ayırmanız gerektiğini gösteren yeni bir yazı tipi de ekledim. Skor yazı tipi love.graphics.newFont alır. Tam olarak aynı yazı tipi dosyası, ancak ' s 32 piksel büyük, çünkü yazı tipi-- veya Pong'da işlendiğinde skor ekranın ortasında oldukça büyük. Ve böylece yaratıyoruz - şimdi iki farklı yazı tipimiz var. Biri mesajımızı iletmek için, diğeri puanımızı göstermek için. Ve sadece bu iki değişkeni, oyuncu bir skoru ve oyuncu iki skoru oluşturacak. Ve sonra dikdörtgenler, soldaki ve sağdaki kürekler için y değerlerimizi başlattık. Y pozisyonlarını takip etmeliyiz, çünkü Pong'daki kürekler sadece yukarı veya aşağı hareket edebilir. Böylece 1Y oyuncusu, dikdörtgeni başlattığımızda, ekrana çizdiğimizde daha önce aldığı değeri alır. 30 yaşında başlayacak, oldukça yüksek. Ve oyuncu 2y oldukça düşük, sanal yükseklik eksi 50, 432 eksi 50 başlayacak. Ve böylece aşık. Güncelleme, Bu, 75 numaralı satırdaki güncelleme işlevinin ilk gerçek kullanımımızdır (geçen dt parametresi). Not, LOVE 2D'nin bunu bizim için geçeceğini unutmayın, ancak bunu vermeliyiz - tanımlamamız gerekir içindeki davranış. Love.keyboard.isDown kullanıyoruz ve bu ilk blok için w ve s dizesini geçiyoruz. Buradaki ilk blok oyuncunun hareketidir. Yani, geleneksel olarak, WASD bilgisayarında hareket etmektir - ve bu örnekte, her iki kürek de hareket etmesine izin vereceğiz, bu yüzden sol kürek için w ve s ve sağ için yukarı veya aşağı kullanacağız kürek. Yani love.keyboard.isDown w, yani-- veya şu anda W tuşuna basarsak, 1y oyuncusu kendini alacak, artı kürek hızı çarpı delta zamanı. Yani yukarı doğru hareket edecek. Negatif kürek hızı alacak, bunu delta zamanı ile çarpın ve bunu kürekimizi yukarı kaydırmanın etkisi olacak y değerine ekleyin. Ve eğer s tuşuna basarsak - 82. satırdaki- tam tersi. Y'yi pozitif kürek hızı ile artırmamız gerekir, çünkü hatırlama, y ​​ekseni hareketi - yukarı negatif, aşağı pozitif. Love.keyboard.isDown dizeleri olarak yukarı ve aşağı kullanmamız dışında, sağdaki raketle aynı şeyi yapıyoruz. Ve sonra aşağıda, daha önce verdiklerimize ek olarak şimdi de skor yapıyoruz. 125. satırda, love.graphics.setFont, scoreFont adını verdiğimizi unutmayın, çünkü bunu çağırmazsak, son yazı tipi ne olursa olsun kullanır, bu varsayılan olarak sekiz piksel yazı tipidir. programımızın en iyisi. Puan yazı tipine ayarlamak istiyoruz, ve sonra love.graphics.print olarak adlandırmak istiyoruz. Bu durumda, sadece printf kullanarak değil, somut yerlere yazdırıyorum. Sanal genişlik iki eksi 50'ye bölünür. Penceremizi nasıl ölçeklendirirsek değiştirelim, pencerenin ortasının solunda her zaman 50 piksel, pencerenin ortasının sağında 30 piksel olacaktır. oyuncuya iki puan kazandırıyor. Ve böylece Pong 3'e girip çalıştırırsak, eskisi gibi görünüyor. Ekranın ortasında sıfır ve sıfır olarak bir skorumuz olduğunu unutmayın. Ama daha da önemlisi, küreklerimizi yukarı ve aşağı hareket ettirebiliriz. Ama bir sorun var, bu yüzden ekranın kenarından öteye gidebilirim, bu da uygulamamızda istediğimiz davranış değildir. Yani bazı etkileşimlerimiz var, ilerliyor ama maalesef daha gidecek çok yolumuz var. Ya da neyse ki. Hadi gidip top güncellemesine bakalım. Yani küreklerimiz var, hareket edebiliyorlar, ekranın kenarının ötesine geçebiliyorlar, ama bir topumuz yok - sadece ekranın ortasında oturuyor. Ve aradığımız şey bu değil. Kürekler arasında gerçekten zıplayabileceğimiz bir top istiyoruz, böylece sadece hareketli küreklerin ötesine geçen gerçek bir oyun elde edebiliyoruz. Bu yüzden bakacağımız birkaç önemli fonksiyon. Buraya ilk bakışımızı rastgele yapacağız. Bu nedenle oyunlarda rastgele sayı üretimi çok yaygın bir şeydir, böylece oyunumuzun farklı örnekleri arasında tahmin edilemezlik ve değişkenlik elde ederiz. Sadece Lua'ya ait önemli bir işlev. AŞK 2D bir şey değil, sadece bir Lua şey. Matematik rasgele tohum uyuşması. Birçoğunuz muhtemelen tohum gibi, rastgele bir sayı üreticisi gibi, tohum ve bu sadece rastgele bir sayı üreteci anlamına gelir. Sahte rasgele olduğundan, tüm rastgele sayılarını temel almak için bir çeşit başlangıç ​​değerine ihtiyaç duyar. Bir başlangıç ​​numarası alır, daha sonra oyun motorumuzda kullanabileceğimiz yeni rastgele değerler elde etmek için bu sayı üzerinde bazı matematiksel işlemler gerçekleştirir. Ancak, her seferinde aynı sayıyı verirsek, bize her seferinde aynı rastgele sayıları verir, yani rastgele olmayacaktır. Çok tutarlı olacak. Bu yüzden rastgele sayı üretecimizi tohumlamak, farklı bir başlangıç ​​değeri veya tohum vermek için bir yola ihtiyacımız var ve bunu math.randomseed somenumb işlevi ile yapacağız. OS.time bunun bağlamında önemli bir işlevdir, çünkü uygulamanızı her çalıştırdığınızda farklı bir sayı almanın çok yaygın bir yolu, geçerli saat kaç saniyede olursa olsun onu geçiriyor, çünkü genellikle oyununuzu her çalıştırdığınızda farklı olacak çok büyük bir sayı, hayır ne önemli. Çünkü çoğu motor bağlamında, Lua bağlamında, Unix Epoch zamanı olarak adlandırılan, sıfır sıfır UTC 1 Ocak 1970, her bir saniyede değişen çok sayıda dokuz veya 10 basamak uzunluğunda. Ve sonra tüm bunlardan gerçekten yararlanmak için, rastgele bir sayı elde etmek için bir fonksiyona ihtiyacımız var ve bu yüzden bunu bir teknikle geçmeniz gerekmese de, bir min ve maks alan matematik.random ile yapıyoruz. min, min'i geçmezseniz min'i dolaylı olarak erteler. Ve o' bu aralık dahilinde bir değer döndürür. Yani math.random bir, 50 derseniz, bize 1 ile 50 arasında rastgele bir rastgele verir. Ve eğer math.random 50 dersek, aynı şeyi yapar. Diyelim ki - rmin'in bir olduğunu varsayar ve bize bir ile 50 arasında bir değer verir. Ve sonra çok temel ama hemen hemen her yerde oyun bağlamında yardımcı olan iki önemli matematiksel fonksiyon, sadece math.min, iki değerden küçük olanı döndürür ve iki değerden büyük olan math.max değerini döndürür. Ve bunu değerleri bir aralığa kenetleme bağlamında göreceğiz. Şimdi bir demoya bakalım. Bu yüzden devam edip Pong 4'ü açacağım. Ve main.lua'ya bakacaklar. Yani burada 47. satırda, daha önce olduğu gibi math.random tohum fonksiyonunu çağırdığımızı görüyoruz. Ve unutmayın ki biz başka bir işlev çağrısı OS.time geçiyor, çünkü bizim uygulama her çalıştırdığımızda OS.time farklı olacak. Bu yüzden, her saniye çalıştırdığımızda başvurumuzu mevcut saniyenin sıfıra, sıfıra, sıfıra, sıfıra, sıfıra, sıfıra, sıfıra, 1 Ocak 1970'e göre ne olursa olsun ekiyoruz. Aynı saniye içinde çalışmadığımızı varsayarsak. Ve sonra 71 ve yedinci çizgiye inersek - ya da özür dilerim, 67 ve 68, şimdi var-- topumuza bir başlangıç ​​değeri veriyoruz, çünkü aslında topumuzu manipüle etmeye başlamak istiyoruz. Bu yüzden ona bir X ve bir Y veriyoruz, bu yüzden onu tekrar merkeze yerleştiriyoruz, ancak şimdi, sadece love.graphics.rectangle işlevimizle statik olarak oluşturmak yerine, bunun için bir değişken tanımlıyoruz. bunun zaman içinde değişmesini istiyorum. Topumuzun ekran etrafında hareket etmesine izin vermek istiyoruz. Yani bu x ve y değişkenleri şimdi değişmeye başlayacak ve mevcut hızına göre değişecek. Ve hızı ball dx ve ball dy'de depolanacak. dx ve dy, delta x ve delta y için ortak kısayollardır, bu da hızı temsil edersiniz. Yapacağımız şey, etkili bir şekilde, delta x ve delta y'ımızın ne olduğunu almak ve bunları kare kare topumuza eklemek ve topumuzun konumunu bir değere göre güncelleme etkisi olacak. Ve deltayı x ve deltayı y ayırmak, topumuz için farklı açılara, farklı yörüngelere sahip olmamızı sağlayacaktır. Ve sonra bu uygulamada yaptığımız başka bir şey, bir oyun durumu kavramıyla başlıyoruz. Çünkü şimdi bir başlangıç ​​durumuna sahip olabiliriz, ve sahip olacağımız şey bir oyun durumu. Bu örnekte ve bu uygulamada yapacağımız tek şey bir dize olarak başlangıç ​​durumu. Gelecek örneklerde, devlet makinesi olarak adlandırılan şeyi kullanacağız ve aslında farklı devletleri kendi modüllerine ayıracağız. Ancak bu oyun bağlamında, sadece nasıl çalıştığını göstermek için basit bir dize kullanacağız ve oyuna başladığımızda ilk durumumuzun başlangıçtaki başlangıç ​​dizesi olması gerektiğini söyleyeceğiz. durum. Ve böylece 86. satırda yapacağımız şey, son örnekte sahip olduğumuz bir sorunu çözmek, yani kürekler ekranın kenarlarının ötesine geçebilir, bu da izin vermemiz gereken davranış değildir. Yani math.max'ı sıfır olarak arayacağız ve daha önce yaptığımız aynı işlemi, ve bu iki değerden hangisinin daha büyük olduğunu döndürme etkisi olacaktır. Eğer değer şuysa, y değerine negatif kürek hızı ekliyor ve negatif aralığa giriyorsak, ekranın üst kenarının ötesinde olduğu anlamına gelir, sıfır bu değerlerden daha büyük olacaktır ve bu yüzden bu durumda her zaman sıfır olacaktır. math.max iki değerden daha büyük olanını döndürür. Böylece onu asla üst kenarın üzerine çıkmayacak şekilde kenetleme etkisi olacaktır. Tersi, sanal yükseklik eksi 20'de math.min ve oyuncu 1.y artı raket hızı olarak adlandırdığımız 96. satır için geçerlidir. Ve bu aynı etkiye sahip olacak, bu iki değerden hangisi daha azsa geri dönecektir. Bu durumda, ekranın alt kısmındaki raketimizin boyutuna göre değiştirilen sanal yükseklik eksi 20'nin üzerine çıkarsak, eksi 20 sanal yüksekliğine ayarlayacaktır. Bu yüzden asla bu noktanın altına inmeyiz. Aynı şeyi ikinci oyuncu için de yapıyoruz, tamamen aynı mantık. Ve eğer oyun durumundaysak, topumuzun pozisyonunu gerçekten güncelleyeceğiz. Yani biz ... başlangıç ​​durumundaysak top hiç hareket etmeyecek. Ama eğer oyun halindeysek, top x'in top x'e eşit olması ve top x delta zamanına eşit olmasını istiyoruz. Ve Lua'da kendi kendine değer eklemek için bir kısayol olmadığını unutmayın, bu yüzden top x'i top x'e eşittir, artı top x çarpı delta zaman, sadece top x artı top x çarpı delta zamanına eşittir. Sadece verdikleri bir dil kararı. Ama eğer oyun halindeysek, bunun şu anki topumuzun hızı ne olursa olsun ölçekleme etkisi olacak ve-- delta zamanı, böylece kare hızından bağımsız kalır. Ve sonra onu x topuna ve y topuna ekler. Ve biz aslında burada 170-174 çizgisinde çalışıyoruz. Şimdi - sadece ekrana düz sayılar yapmak yerine, aslında x topu ve y y topunu kullanıyoruz. Ve eğer oyun durumundaysak, bunlar güncellenir. Ama eğer 127. satıra geri dönersek, şimdi aşka girdik. Tuşlu fonksiyon, bu yüzden 120. satıra başlıyoruz. Eğer anahtarın kaçmasına eşit olsaydı, önce love.event.quit. Ancak şimdi 127. satırda, anahtarın giriş veya dönüş için eşit olup olmadığını kontrol edeceğiz ve sonra bunu sadece durum değişikliklerini test etme yöntemimiz olarak kullanacağız. Bu yüzden oyun durumunun başlangıca eşit olup olmadığını söyleyeceğiz, girdiğinizde, oyun durumunun oynamaya eşit olması gerektiğini söyleyeceğiz. Aksi takdirde, yeniden başlatmak için geri ayarlayın. Ve tekrar başlayacağız, x ve y merkezimizde olacak şekilde yeniden başlatacağız, sanal genişlik iki eksi ikiye bölünmüş, sanal yükseklik iki eksi ikiye bölünecek ve vereceğiz yine ilk rastgele başlangıç ​​hızı. Ve burada, bu matematik. Rasgele iki eşittir-- bire ve 100'e veya negatif 100'e eşittir. Bu sadece Lua'nın üçlü bir işlem yapmanın yolu. Yani C'de sık sık sahip olacaksınız-- matematik gibi bir şey olacaksınız. İki tane bir eşittir ve bir soru işareti, 100 iki nokta, negatif 100 olurdu. Aynı şey, ama Lua buna sahip değil üçlü bir operasyon için bir tür steno, bu yüzden ve ve veya ile yapıyoruz. Aynı şeyi yapmak için mantıksal işlemleri kullanıyoruz. Ve burada da o matematiği sergilediğimizi unutmayın. random bir argüman ya da iki argüman alabilir. Bu durumda, math.random iki diyoruz, bu da bize bir ile iki arasında bir değer vereceği anlamına geliyor. Yani 50-50. Ve sonra negatif 50-50 yaparsak, negatif 50 ve 50 arasında bir değer elde edeceğiz demektir. Yani 100 aralığı, etkili bir şekilde. Ve bunun ne etkisi var, eğer uygulamamızı çalıştırırsak, Pong 4'e gidiyoruz. Biz başlangıç ​​durumundayız, şimdi render ediyoruz - eğer başlangıç ​​durumundaysak, bu mesaj. Enter'a basarsak, top rastgele bir hız alır - aslında hızı kare kare uygular. Güncelleme yönteminde güncelleniyor. Tekrar Enter tuşuna basarsak, sıfırlanır ve başlangıç ​​durumuna geri döneriz. Tekrar yapıyoruz, rastgele bir değer kazanıyor. Tekrar yap, rastgele değer. Rastgele değer. Her seferinde farklı bir rastgele top değeri elde ediyoruz. Ama gerçekten çalıştırmaya çalışırsak veya onunla etkileşime girmeye çalışırsak ne olur? Hiçbir şey değil. Doğruca geçiyor. Dolayısıyla, oyun motorumuzun temel bileşenlerini uygulamamıza rağmen, somut bir oyunumuz yok, hiçbir şey etkileşime girmiyor. Ve bu bakmamız gereken önemli bir parça. Ve bir sonraki - bunu yapmaya başlamadan önce, yine de, sınıf güncellemesine, Pong 5'e bakacağız. Ve böylece daha fazla girmek için - kodumuzu daha fazla ölçeklendirmek için etkili bir şekilde, sınıflar açısından bakmaya başlamamız gerekir. Ve topumuz için bir x ve ay, kürek için bir x ve ay yerine, bir delta x, topumuz için bir delta y, kodumuzu şişirmeye başlayan her yerde farklı değişkenler, onunla çok çıldırmadan önce, sadece küreklerimiz veya top nesnemiz açısından düşünebilmemiz için bu verileri nasıl bir araya getirebileceğimizi düşünmeliyiz. Ve böylece sınıf denen şeyi kullanıyoruz. Bilmediğiniz bir sınıf, şu ana kadar kullandığımız tüm bu değişkenleri almanın bir yoludur, ancak bunları bir kapta bir araya getirerek paddle.x veya raket diyebiliriz - bilirsiniz, bu durumda , araba. Eğer sürücü araba denilen bir fonksiyonumuz varsa, şimdi bunun yerine car.drive diyebiliriz. Değerlerimizden ayrı fonksiyonlara sahip olmak zorunda değiliz, yani-- hepsini bir araya getirebiliriz. Tüm bu farklı değişkenleri her yerde kullanmak yerine, arabanızın mevcut kilometresinin ne olduğunu sorabiliriz. Yani sınıflar etkili bir şekilde planlardır. Kullanın - bir sınıf tanımlarsınız. Diyorsun ki, tamam, benim araba sınıfımda bir-- olacak ' bir kilometre değişkeni olacak, bir boya değişkeni olacak, bir marka ve bir modeli olacak, tüm bunlara sahip olacak ve kendi durumunu koruyacak. Bunların hepsini bizim için koruyacak. burada görüldüğü gibi. Ve tipik olarak, bunlara alan denir. Ve sonra yöntemlerimiz de olacak. Bu verilerden tamamen ayrı olmak yerine, bir otomobilin artık temel olarak kendi işlevleri vardır. Sürücü veya dönüş veya honk, et cetera adlı kendi yöntemine sahiptir ve arabayı çevir veya honk araba, et cetera gibi bir işleve sahip olmamız gerekmez. Ve sonra bu sınıf etkili bir plan. Kısaca bir sınıfı nasıl tanımlayacağımızı göreceğiz, ancak aslında kendi veri kümesine sahip bir kürek ve kendi veri kümesine sahip başka bir kürek gibi, tanımlamamız lazım-- bu sınıftan nesneler oluşturmalı, başlatmalıyız. Temel olarak, bu sınıfı bir plan olarak kullanın, ancak bir fabrikaya götürün ve plandan somut arabalar oluşturun. Ve bunlar nesneler. Ve burada görüldüğü gibi, küreklerimiz ve topumuz bunu yapmak için mükemmel basit kullanım durumlarıdır. Öyleyse devam edelim ve Pong 5'e bakalım. Yani Pong 5'de, hemen dizin yapısına bakarsanız, bir ball.lua ve paddle.lua eklediğimizi görebilirsiniz. Sınıf isimlerini büyük harflerle yazabilmeniz için nesne isimlendirilmiş programlamaya sahip dillerin çoğunda, sınıfları somut nesnelerden, değişkenlerden veya fonksiyonlardan ayırabilirsiniz. Eğer main.lua'mıza giderseniz, 35. satırda, sınıf adı verilen bir kütüphane istiyoruz. bu sınıfları yaratmamıza izin verecek. Sınıflar yerel olmadığından, bir anlamda yerel bir Lua özelliğidir, ancak Lua'nın nesne yönelimli programlama yapma şekli biraz kıvrıktır. Bazı insanlar, onu daha basit hale getiren ve Java veya C # veya hatta Python gibi nesne ağırlıklı programlamayı daha ağırlıklı yapan diğer dillerle çok daha yakından ilgili bir kütüphaneyi bir araya getirdiler. bu kütüphanelere çok benzer. 39 ve 43. satırlarda, kendi kodumuzu, kürek ve topumuzu alıyoruz ve şu anda bunlara bir göz atacağız, böylece bir sınıfın neye benzediğini görebileceğiz. Bu yüzden devam edeceğim ve bir-- top dosyasını açacağım, ball.lua. Ve burada bir top sınıfı oluşturmak için tek yapmamız gereken, sınıf kütüphanemizi kullanarak top sınıf alır, ve sonra böyle kıvrımlı parantezler. Ve şimdi bir sınıf nesnesimiz, bir sınıf tablomuz var, çünkü Lua'daki her şey bir tablo. Ama bunu nesneler açısından düşünebiliriz. Top adında bir sınıf nesnesimiz var ve sonra bu sınıfa ait fonksiyonları tanımlamaya başlayabiliriz. Bu yüzden yapıcı veya init fonksiyonu olarak adlandırılan bir başlatıcı olarak adlandırılacak olanı tanımlayacağız. Ve topumuzu istediğimiz şeyle başlatmamıza izin verecek. Bu durumda, topumuza x ve ay, genişlik ve yükseklik ile başlamak istiyoruz. Ve içeride dikkat, benlik denen bir kelimemiz var. Benlik ve bu nesne yönelimli programlama dillerinde yaygın olarak kullanılan kelimelerdir, bu da bu sınıfla yarattığımız her nesne benlik olacağı anlamına gelir. Yani bunu göreceğiz-- kısa süre içinde göreceğiz. Self.x x alır. Bu çağrıyı kullanarak oluşturduğumuz somut nesne ne olursa olsun, bu başlangıç ​​çağrısı, x'ini bu x'e, y'sini y'ye, genişliğini, yüksekliğini ayarlayın. Bu özel nesne. Öz. Ve sonra delta y ve delta x için aynı şeyi yapıyoruz, sadece bu iki rastgele değeri daha önce yaptığımız gibi ayarlıyoruz. Self.dy, self.dx. Bu, kodda göreceğimiz gibi, bu init çağrısını kullanarak belirli bir nesnenin somutlaştırıldığı her şeye aittir. Biz sadece kolaylaştırmak için burada sadece bir sıfırlama fonksiyonu tanımlamaktayız. Topumuzu ekranın ortasına yerleştiren ve rastgele bir hız veren birkaç kod satırından önce. Bunu şimdi yapıyoruz ve bu, mantık gruplarını yeniden düzenlemenin iyi bir yoludur. Tüm bunları tek bir işlev çağrısında yapan reset adında bir işlev yaratıyoruz ve bunu ana işlevimiz dahilinde çağırıyoruz, kodumuz yoğunlaştırılıyor. Ve şimdi bir güncelleme ve oluşturma fonksiyonumuz olduğunu fark edin. Ve bunları kendi güncellememizden ve kendi çizim fonksiyonumuzdan arayacağız, böylece oyunumuzda, her varlıkta istediğimiz her nesne ve gelecekte bu oyunu oyun üzerine inşa edeceğiz. Sadece güncellemeyi çağırıp main.lua'mızdan her şeyi işleyeceğiz ve hepsini her bir sınıfa ve nesneye erteleyeceğiz, böylece 800. satırlık bir main.lua'ya sahip olmamız gerekmiyor. Burada topla ilgili tüm güncellemeleri ve burada topla ilgili tüm render kodlarını dağıtıyoruz, her topun güncellenmesini ve renderlenmesini çağırıyoruz ve yeniden düzenleme konusunda kendimize çok zaman kazandırıyoruz. Rakete bakarsak aynı şeyi yapıyoruz. Raket de bir sınıf. Sınıf kütüphanesinin sebebini alır. Burada aynı şey, yükseklik xy ve bir boya. Bu durumda, hareket etmemek için bunu sıfıra indiriyoruz. Ve sonra güncelleme fonksiyonunu buraya çağırıyoruz. Yani eğer boyamız sıfırdan küçükse, daha önce olduğu gibi, ekranın üst kenarı ile math.max işlevini kullanıyoruz ve sonra bizim y artı mevcut dy'imiz ne olursa olsun, delta y. Ve sonra burada, self.y math.min, sanal yükseklik eksi self.height, self.y artı self.dy kez delta zamanı alır. Bu, daha önce kürekler ile gördüğümüz sıkıştırma davranışı, sadece şimdi, anadan aldık ve güncelleme fonksiyonumuza koyduk, böylece her kürek güncellemesini çağırıyor ve ana dosyamızdan bazı kod satırları alıyoruz. Ve sonra burada kürek gibi kendi render işlevi var. Raket için render fonksiyonu ve top için render fonksiyonu etkili bir şekilde aynıdır. Ve böylece ana bölümümüze gidersek, biz-- raketi alıyoruz ve onları kullanabilmemiz için topu alıyoruz. Yani eğer top dx ya da top dy, top x, top y, raket y, oyuncu 1y, oyuncu 2y'yi başlatmak yerine 79 numaralı satıra gidersek, şimdi bir tane var, 10, 35, 20. oyuncu iki kürek, sanal genişlik eksi 10, sanal yükseklik eksi 30, 520. Top bir top, sanal genişlik iki eksi iki, sanal yükseklik eksi iki, eksi iki, [? beşe?] iki eksi iki, dört ve dört. Bu kürekler artık kendi x ve y, kendi genişlik ve yükseklikleri üzerinde ve savaşta - veya top xy genişliği ve yüksekliği üzerinde kendi kontrolüne sahip. Ve benlik bu nesneye uygulanır. Yapıcımızda daha önce gördüğümüz şey buydu. Bu yüzden şimdi bile sadece arayabiliriz - bu yüzden oyuncu 1.x oyuncu 1. genişlik, oyuncu 1.y ve her şey bulunur. Oyunumuzda olup bitenleri takip etmek için bir milyon değişkene ihtiyacımız yok. Ve bu, ölçeklendirdikçe özellikle önemli olacak ve elimizde - belki bir seferde ekranda 100 şey var. X'in 100 olduğu x değişkenini istemiyoruz, ancak izlememiz gereken birçok özellik var. Tüm bunlar aynı mantık dışında, şimdi, güncelleme fonksiyonumuzda oyuncuya bir güncelleme ve oyuncu iki güncelleme çağırıyoruz, burada tüm bu mantığa sahip olmak yerine, nerede hareket ettiklerini takip edip gitmediklerini takip ediyoruz ekranın üst ve alt kenarlarından Ve eğer oyun durumu oynarsa, biz şimdi sadece top güncelleme çağırıyoruz. Ve bunların hepsi delta zamanında geçiyor. Ve sonra aynı şey burada. Topu bir kod bloğu olarak yeniden düzenlemek için tüm bu mantığa sahip olmak yerine, çıkardık, yeniden düzenledik, top sınıfımıza koyduk ve şimdi tek yapmamız gereken tek bir kod satırı, top veya sıfırlamak . Ve sonra burada çekme fonksiyonumuzda 169. satırda, sadece bir oyuncu, iki oyuncu, top render var. Ve daha sonra ölçeklendirdikçe ve ekranda çok daha fazla şey, çok daha fazla varlık olan oyunlar ürettikçe, bu renderleri bir döngü içinde yapabiliriz. Ekranımızdaki her varlık için söyleyebiliriz, sadece render edin. Ekranımızdaki her varlık için güncelleyin. Güncelleme mantığını erteleyerek ve her bir varlığa mantık oluşturarak binlerce-- yüzlerce kod satırını sadece birkaç kod satırına yoğunlaştırabiliriz, nesne yönelimli programlama sayesinde. Ve böylece sınıfları kullanarak yeniden düzenleme yapacağız. Peki, bunların şu ana kadar nasıl çalıştığıyla ilgili sorularınız mı var? Güzel. Bence bu beş dakikalık bir mola vermek için iyi bir nokta. Ve geri döndüğümüzde, saniyedeki karelere nasıl bakacağımız hakkında konuşacağız. Tamam. Bu yüzden sadece küçük bir şeye bakmak için bir dakikanızı ayıracağız, ancak çoğu zaman oyunlarımızda iyi performans gösteren uygulamalarımız gibi performans gösterdiğimizden emin olmak istiyorsak, bir şekilde ... saniyedeki kare sayısını izleyin. Ve böylece, bunu hızlı bir şekilde açıklamak için bir saniye alacağımı düşündüm ki gelecekte bunu kullanabilelim. Burada bizim için önemli olacak iki işlev - bunların birincisi, uygulamaya biraz küçük bir kozmetik ilavesi. Sadece aşk. Pencere. setTitle. Başlık, şimdiye kadar başvurumuz, varsayılan olarak ne dediğinden emin değilim, sanırım diyor - ne diyor - başlıksız. Evet. Yani bu-- az ya da çok cila eksikliği tabakası. Ve bu sorunu çabucak çözmek güzel olurdu. Bu yüzden, bu sorunu hızlı bir şekilde çözecek olan love.window.setTitle adında bir işlev çağıracağız. Bu ayrıntıya sahipmiş gibi görünebiliriz. Ve sonra aslında iyi çalışıp çalışmadığımızı veya çok kötü çalışıp çalışmadığımızı belirlememize izin verecek olan şey, love.timer.getframespersecond.getFPS adlı bir işlevdir, ki bu SEVGİ'nin bize ücretsiz olarak verdiği ve bize izin verdiği bir şeydir. İstediğimiz yere kolayca tokat atmak. Konsola yazdırabiliriz veya doğrudan uygulamamıza çizebiliriz. Bu durumda biz ikincisini yapacağız. Bu yüzden devam edip bizim ana yolumuzda AŞK-- ya da Pong 6'ya gireceğim. Devam edip hatta - nerede olduğu - 64. çizgiye bakarsak, love.window.setTitlePong, sadece hızlı ve kolay. Şimdi, pencere başlığımız uygun şekilde ayarlandı ve eğer 198 satırına inersek, burada bunu 198 satırında display FPS adı verilen ayrı bir fonksiyonda ayırmaya karar verdim. Ve fonksiyon 207 satırında tanımlandı, böylece bir işlev ekranı FPS, parametre almaz. Tek amacı mevcut FPS'imizi ekrana çizmektir. Şimdi mevcut yazı tipimizi küçük bir yazı tipine ayarlayacağız. Rengimizi ayarlayacağız - bu yüzden daha önce bahsettiğim şey, LOVE'nin render rengini bazı RGBA dörtlü olarak ayarlayabilmemiz ve bu noktanın ötesinde çizdiğimiz her şey çizilecek - bu renk ne olursa olsun. Yani, Bu durumda, rengimizi tamamen yeşile ayarlama etkisi olan sıfır, 255 yeşil, sıfır mavi, 255 tamamen opak kırmızı veriyoruz. Ve sonra love.graphics.print, şu anki FPS-- dizemiz ve sonra da şu anki FPS'imiz, yani love.timer.getFPS. Ama bunu bir sayı olarak döndürecek ve varsayılan olarak, Lua dizeleri ve sayıları birleştirmenize izin vermiyor, bu yüzden burada Lua'da dize birleştirme yapmanın yolu olan bu ..operator ile birleştireceğiz. İki dize işlevini çağıracağız. Yani aşık olacağız. Tim.FPS, onu bir dize yapacağız ve sonra burada birleştireceğiz. Ve sonra love.graphics.print değerini bu değere çağıracağız ve sonra 10, 10'a koyacağız. Ekranın sol üst kenarından biraz kaydırın. Yani bu Pong 6'ya gitmenin etkisine sahip olacak ve biz de onu yönetiyoruz. Şimdi sıfırdan ve 52'den başlayabildiğini görebiliyoruz, çünkü aslında kullanabileceğimiz bir sayıya sahip olmadan önce birkaç veri çerçevesi toplamak zorunda. Ama orada 60'da FPS görüyoruz ve bu yüzden oyunumuz çalışıyor, aksi takdirde aynı. Tamamen rastgele. Biraz kırılmış, ama sorun değil, düzelteceğiz. Ama şu anda bir sorunumuz var ve bu da topumuzun doğrudan küreklerimizden geçtiği. Peki bu sorunu nasıl çözebiliriz? Çarpışmayı tespit etmenin bir yoluna ihtiyacımız var. 2B oyunlarda, genel olarak, aa bb çarpışma algılama kavramı vardır. Ve bunun ekseni hizalı sınırlama kutusu çarpışma tespiti, yani sınırlama kutularımız var, sadece dikdörtgenler, dörtlüler, x ve ay ve döndürülmemiş bir genişlik ve yüksekliğe sahiptir. Yani eksenlerimizle tamamen uyumlular. Tamamen paralel dikeydirler. Bu yüzden bu kolay matematiği elde etmenin tek yolu, aa bb çarpışma tespiti çalışması, kutularımızın rotasyonunun olmamasıdır. Tamamen hizalanmış olmaları gerekir. Ama eğer öyleyse, çok basit bir algoritmaya sahibiz, yani kutularımızın hiçbir kenarının diğer dikdörtgenin karşıt kenarlarının dışında olmadığından emin oluyoruz. Eğer bir dikdörtgenimiz varsa-- ve bunu burada ekranda göstereceğim. İki dikdörtgenimiz var. Eğer bu üst kenar bu kenarın altındaysa, ne olursa olsun, aralarında birleşmeyeceklerini biliyoruz - kesişmiyorlar. Yapmanın bir yolu yok çünkü burada. Yani x ve y'de nerede olursa olsun, eğer aşağıdaysa, bu bir çarpışma değil. Bu kenar bu dikdörtgenin bu tarafındaysa, bu iki kutunun üst üste binmesinin bir yolu olmadığını da biliyoruz. Ve karşı kenar olduğu sürece her kenar için geçerlidir. Bu kenar bu kenarın altındaysa, bu kenar bu kenarın üzerindeyse, bu kenar sağdaysa ve bu kenar soldaysa, ne olursa olsun, bu kutuların çarpışmayacağı anlamına gelir. Böylece dört koşul yapabiliriz. Rec1.x, rec 2.x değerinden daha büyük değilse, rec2.width ve rec1.x plus rec1.width değerinin rec2.x değerinden az olmadığını söyleyebiliriz, bu nedenle iki kenar zıt kenarların ötesinde değilse, y ve y artı rec1.height ile aynı şey, bir çarpışma olduğunu biliyoruz. Bunu biliyoruz çünkü bu kriterlerin hiçbirini yerine getirmedik. Ama biliyoruz ki bu doğru değilse, - kenarlardan biri karşı kenarın ötesinde değilse, o zaman-- bir çarpışmamız olur. Yani gerçek olacak. Bunu burada kodumuzda göreceğiz. Pongs 7'ye giden hat 113'te top çarpışması olarak adlandırdığımız bir fonksiyonumuz var. Top sınıfımız çarpışmalar adı verilen bir işleve sahiptir. Hadi gidip top sınıfımıza bir göz atalım. Ve bu durumda, fonksiyonumuzu bir kürek parametresi alacak şekilde tanımladık, bu nedenle xy ve genişlik ve yüksekliğe sahip başka bir dikdörtgenle karşılaştırılacak. Ve diyoruz ki eğer rx raket x'den büyükse artı raket genişliği, yani rx sağ kenardan büyükse. Yani sol üst köşemiz-- ya da sadece solumuz - sağ kenardan büyükse, çarpışamayacağımızı biliyoruz. Aynı şey olursa ' s diğer dikdörtgenlerden daha büyük, self.x artı self.width. Hayır, özür dilerim. Bu durumda, raketin x değeri-- temelde aynı işlemdir ancak raketin bakış açısından. Raket sağ taraftaki dikdörtgenden daha büyükse, sağ kenar boyunca sağ kenarı geçecek şekilde daha uzaksa, çarpışma olmayacağını biliyoruz. Sadece imkansız. Y ile aynı şey. Eğer y-- self.y ise, bu top y-- raketin y değerinden ve raket yüksekliğinden daha büyükse. Yani eğer raket kenarının altındaysa, çünkü yüksekliği dikkate alıyoruz veya raketin y değeri bu topun y artı kendinden yüksekse, o zaman bunun da bir çarpışma olamayacağını biliyoruz. Ama bu doğru değilse, o zaman doğruya dönmeliyiz. Ve böylece ana bölgemize geri dönersek-- hayır, bu yanlış ana. Burada main.lua'ya geri dönüyoruz. Biz ball.collides diyoruz. Yani eğer oyun durumumuzdaysak, eğer bizim durumumuzdaysa - üzgünüm, eğer oyun durumumuzdaysak, oyun durumu oyuna eşitse, top bir oyuncu ile çarpışırsa, oyuncu bir sol kürek. Eğer bir çarpışma tespit edilirse, ball.dx ve dx x hızımızdır. Yani x ekseni üzerinde hangi yöne hareket ediyorsa. Yani eğer giderse sola hareket edecek - bir çarpışma tespit edersek. Ve sağa veya sola hareket etmesi önemli değil, ama ihtiyacımız vardı - yapmamız gereken şey onu negatif değerine ayarlamak. Çünkü eğer sola hareket ederse ve diyelim ki - diyelim ki negatif 20 pikselde sola hareket ediyor ve 20'ye ayarlıyoruz, dx şimdi 20, sağa hareket etmeye başlayacak. O' x hızını tersine çevirme ve dolayısıyla yönünü tersine çevirme etkisi olacaktır. Ama burada 1.03 ile de yaptığımız şey, sadece oyunu hızlandırmak için biraz çarpıyoruz. Çünkü oyunun sürekliliğe girmesini istemiyoruz sadece aynı hızda. Heyecanı artırmayacak. İşleri devam ettirmek istiyoruz, biraz ivme kazanmak istiyoruz, bu yüzden yapacağımız şey ball.dx'i negatif değer sürelerine keyfi olarak belirlediğimiz bir ölçekleyiciye eşittir. Bu durumda, puanın 1.03 olması gerektiğine karar verdim, böylece her seferinde% 3 artıracak. Ve sonra bir - topumuz - olması durumunda x eklendiği için x hızı her kareyi konumuna ekleniyor, kürekimizin içine benzemediğinden emin olmak istiyoruz. Çünkü aynı işlem geçerli olduğu için belirli sayıda pikseli sola veya sağa kaydırabilir. Öyle ki ikisi birbirinin üstüne benzer. Yeniden istiyoruz - kaydırmak istiyoruz, sıfırlamak istiyoruz. Şimdi ne yapacağız-- çünkü durum böyleyse başka bir çarpışmayı anında tespit edecektir. Bir sonraki karede, o raketin içindeyse, o raketle hala çarpıştığını söyleyecek, böylece hızını tekrar değiştirecek. Ve bunun etkisi, kürek içinde sonsuz bir şekilde zıplıyor. Bunun olmasını istemiyoruz. Dolayısıyla bir çarpışma tespit edersek, onu değiştirmek istiyoruz. Bunun, raketin çarpışma kutusunun tamamen dışında olduğundan emin olmak istiyoruz. Yani ball.x'in one.x oyuncusu olduğunu söylüyoruz, artı beş. Artı beş çünkü kürek genişliği. Bu, bir çarpışma, negatif set - x hızını negatif olarak tespit ettikten sonra sadece sol küreklerin sağ kenarında sağa kaydırın. Burada da aynı şeyi yapıyoruz. Top bir veya iki oyunla çarpışırsa, x hızını reddediyoruz veya tersine çeviriyoruz. Ve sonra-- bu aynı işlemdir, ancak sol üst sol köşeye dayandığı için, eksi beş kadar ekleyemeyiz, bu mantıklı olmaz. Dörtte eksi yapacağız çünkü topun genişliği bu. Beş tane küçültsek, bir piksel alanımız olurdu. Bu örnekte bunu beşe kadar artırdık, çünkü sağ taraftan geliyoruz. Sadece küreklerin sağ kenarında olduğundan emin olmak istiyoruz, yani one.x oynatıcısına ayarlıyoruz. Ve bu durumda, eksi dört kullanıyoruz çünkü topun genişliği bu. Bu yüzden onu sola, topun genişliğine kaydırmak istiyoruz ve bu sağ kürek etkisine sahip olacak, bir çarpışma varsa, sadece kayacak ve sağa - top olacak kürek iki sağa doğru dokunarak. 118. satırda, daha önce yaşadığımız sorunu - oh, üzgünüm, aslında düşündüğüm şey değil, çözüyoruz. Bu-- eğer bir çarpışma varsa, o zaman topun hızının her seferinde rastgele olmasını istiyoruz. Bu, oyunu oynarken oynadığımız zamanın etkisi var ve iki kürek arasında bir çarpışma tespit ettik, Her seferinde aynı açıyı ileri ve geri almak istemez, çünkü o zaman oyun sonsuza dek aynı şekilde gerçekleşir - aynı açı tekrar tekrar olmaya devam edecektir. Biz bunu yapmıyoruz. Topun raketten nasıl sektiği konusunda biraz değişkenlik istiyoruz. Yani bu, hala koşulda, eğer top bir oyuncu ile çarpışırsa, diyelim ki, topun y hızı negatifse, o zaman negatif olmaya devam etmek istiyoruz. Hâlâ topu istiyoruz - sanki top bir çeşit yukarı doğru açıyla geliyor ve raketten sekiyorsa, x hızının değişmesini istiyoruz. Bunun ters yöne gitmesini istiyoruz, ancak topun yukarı çıkmasını istiyoruz. Topun geri sıçramasını istemiyoruz, bu hiç mantıklı değil. Y hızını reddetmek istemiyoruz. Yani biz ' y hızını negatif tutacağız, 10 ile 150 arasında negatif bir değere ayarlayacağız. Ve bu sadece keyfi. Bunu istediğiniz gibi ayarlayabilirsiniz. Ve eğer y hızı pozitifse aynı şeyi yapacağız. Topun gitmesini istiyoruz - eğer zaten aşağı iniyorsa, topun pozitif yönde gitmesini istiyoruz. Burada da aynı şeyi yapıyoruz. Oyuncu iki örneğinde aynı mantık. Ve sonra bir saniyeliğine daha önce baktığımı düşündüm, ancak ekranın üst ve alt sınırını bu şekilde düzeltiriz. Sağ. Çünkü küreklerin şimdi topu saptırdığı gerçeğini çözmek için bir şey var, ancak topun ekranın üst kenarından veya ekranın alt kenarından sonsuz olarak geçmesini istemiyoruz. Yani bu sadece basit bir durumdur. Sadece topun sıfıra eşit veya daha az olup olmadığını söylüyoruz, yani top ekranın üst kenarındaysa, sıfıra ayarlayın, bu yüzden ekranın kenarının üzerine çıkmadığından emin olun ve daha sonra geniş hızını ortadan kaldırır, böylece anında aşağı doğru gitmeye başlar. Evet. İZLEYİCİ: Bu soru Pong 7, satır 113 hakkındadır. Bir çarpışma varsa dx ve y toplarının çarpışması işlevinde yer değiştirmesi yapılamaz mı? COLTON OGDEN: Top çarpışırsa topun işlevinin değiştirilmesi - hayır, topu çarpışırsa - çarpışmalar işlevi a-- sadece doğru veya yanlış döndürür. Yani-- Yani, sanırım, muhtemelen bu şekilde yeniden düzenleyebilirsiniz, ama çarpışmanın amacı böyle bir yan etkiye sahip olmak değildir. Tek amacı sadece doğru ya da yanlış döndürmektir. Çünkü her şeyi yapabiliriz - istediğimiz her türlü davranışa sahip olabiliriz. Bir çarpışma işlevinde, topu değiştirmek veya herhangi bir şey yapmak istemeyebiliriz, sadece doğru dönmesini ve konsola bir şey yazdırmasını isteyebiliriz. Yani, bence, mühendislik perspektifinde, basit bir doğru veya yanlış fonksiyona sahip olmak daha mantıklıdır ve daha sonra oyun durumunuzu aslında ana fonksiyonunuz içinde veya başka bir şeyin içinde nasıl etkilemek istediğinizi belirleyin işlevi. TAMAM. Ve böylece, Evet, buraya geldik. Ekranın üst kenarı ve ardından ekranın alt kenarı. Eğer top.y, aynı şey, ekranın sadece alt kenarı. Ball.y sanal yükseklik eksi dörtten büyük veya ona eşitse ve sanal yüksek eksi dört yapıyoruz, neden? KİTLE: [Duyulmuyor] COLTON OGDEN: Kesinlikle. Bu yüzden yazdığımızdan emin olmak istiyoruz - en kısa sürede - topun alt kenarı ekranın altına değiyor, bir çarpışma tespit etmek istiyoruz, sonra topu söylemek istiyoruz. Y ,-- sanal yükseklik eksi dört zamanın ne kadar sürdüğüne ve hızın ne kadar olduğuna bağlı olarak alt kenarı aşması durumunda, anında hemen yukarı doğru yerleştirmek istersiniz, böylece temiz bir sıçrama. Ve sonra, y hızını yukarıda yaptığımız gibi reddetmek istiyoruz. Ve eğer burada programımızı çalıştırırsak, Pong 7 aynı görünüyor, ama şimdi top zıplıyor. Ve bir negatif var - rastgele var - alt kenarın altına gidiyor gibi görünüyor çünkü monitör şu anda 720'de ve bu pencere çözünürlüğü, ancak alt kenardan da sekiyor. Ve açı, eğer dikkat ederseniz, her seferinde biraz farklıdır, çünkü ona rastgele bir y hızı, bir y-- evet veriyoruz. Ve sonra bu etkiliyor-- ah, berbat ettim. Hız artışını göstermek istedim. Biraz zaman alacak. Ancak her çarpışmayı tespit ettiğinde, x hızını 1,03 oranında artıracaktır. Yani biraz daha hızlı olacak. Şimdi, y açısı biraz dik, bu yüzden bunu göstermek sonsuza kadar sürecek. Ama bunu daha sonraki bir örnekte göreceğiz. Oyunumuzun temelleri var. Ama skoru nasıl koruyoruz? Pong'da skoru nasıl tuttuğumuzun belirleyici faktörü nedir? Sol veya sağ. Ekranın sol veya sağ kenarından geçtiği sürece. O halde ne yapmamız gerekiyor? İZLEYİCİ: [duyulmuyor] COLTON OGDEN: Bir sayaca ihtiyacımız var, ayrıca topun ekranın sol veya sağ sınırıyla çarpışıp çarpışmadığını da izlememiz gerekir. Ve sonra o sayacı artırın. Bu yüzden bunun nasıl uygulandığını görmek için Pong 8'e bir göz atacağız. Burada 88 ve 89 satırlarında bazı sayaç değişkenleri, oyuncu bir skoru, oyuncu iki skoru var. Bunları uzun zamandır yaşadık, ama kullanmadık. Onları sadece ekrana çizmek için kullandık. Şimdi onları arttıracağız ve buradaki kodumuzda puanlama değişkenleri olarak göstereceğiz. Pong 8'de uyguladığımı sanıyordum, ancak puanın gerçek artışını dışarıda bırakmış olabileceğimi düşünüyorum. Fakat bu, bu örnekle ilgili mantık. Yani ball.x sıfırdan küçükse, bu sadece ekranın sol kenarından geçtiysek, şimdilik servis oyuncusunu görmezden gel. Önemli olan şu anda oyuncu iki skor yapıyoruz, oyuncu twp skoru alıyor, artı bir tane. Sadece basit bir artış. Ve sonra topu sıfırlıyoruz. Burada da aynı şey var. Ball.x sanal genişlikten büyükse, ekranın sağ kenarından geçin ve gerçekte olabilir - eğer top x artı dört sanal genişlikten daha büyükse, o zaman aynı etkiye sahip olacaktır. Ama aslında hayır, çünkü gol attığında topu hiç görmediğimizden emin olmak istiyoruz. Yani, evet, bu aslında doğrudur. Ball.x sanal genişlikten büyükse, servis atan oyuncu iki olur, oyuncu bir puan oyuncu bir puan artı bir olur. Ve sonra topu sıfırlayacağız. Servis oyuncu. Şimdi konuşmamız gereken şey hizmet etmek fikri. Oyuna başladığımızda-- hadi gidip bir göz atalım-- Şimdi Pong'a gidiyoruz, bu yüzden doğrudan Pong 9'a gideceğiz ve sonra bir devlet makinesinin ne olduğuna bir göz atmamız gerekiyor. Şu anda oyunda biraz devletten bahsettik. Başlangıç ​​durumuna geçtik, bu da oyunun Enter tuşuna basmamız için hazır olduğu ve top rastgele bir yönde gideceği anlamına geliyor. Ve sonra oyun durumumuz var. Ve oyun durumu topla etkileşime giren küreklerimize ayarlanır ve daha sonra skoru takip eder. Durum makinesi çok önemlidir. Oyun geliştirmede her yerde bulunan bir konsept. Bunun anlamı, yeni devletleri ortaya çıkarmak için hangi eyalette olduğumuzu ve bu eyaletler arasında hangi geçişlerin gerçekleştiğini nasıl izleyebileceğimiz anlamına gelir. Ve her bir devletin kendi mantığı vardır. Ve bu devletlerin mantığını ayrı ayrı kırarak, kodumuzu çok daha büyük ölçeklendirebiliriz ve monolitik koda sahip olmayabiliriz - bu özel şema, Mario gibi bir karakter için durum makinesi, aşağı inen bir serbest bırakma durumunuz olan bir durum makinesi olarak sahip olabileceğinize bir örnektir. - in - aşağı girişi gibi. Yani eğer serbest bırakırsak, ayakta kalacaktır. Ördek durumu, geçiş aşağı anahtarı serbest bırakıyor, ayakta duruyor. Ayakta tuşu, aşağı tuşuna basın. Ördekleşiyor, bunlar devletler ve geçişler. Bu tek tek devletler, genel olarak davranışlarının genel temsilidir. Aynı mantık oyunumuz için de geçerli. Bir oyun durumumuz var, bir hizmet durumumuz var. Belki devlet üzerinde bir oyun oynamak istiyoruz. Birisi 10 puan alırsa, o zaman diyelim ki, kazanan x. İstediğiniz sayıda durumu tanımlayabilirsiniz, Bu - hangi modele bağlı olursa olsun geliştirmek istediğiniz oyun. Örneğin, Super Mario'nun bir başlık ekranı olduğu gibi, belki de oyununuz yüksek bir skor durumuna sahip olabilir. Oyununuzdaki tüm yüksek puanları görüntülemek istiyorsunuz ve bunu önümüzdeki hafta bir derste göstereceğiz. Fakat bu bir devlet makinesidir. Bu sadece bir seferde herhangi bir belirli durumda olabilir ve geçişler, eyaletleriniz arasında girmenize izin veren şeydir. Ve her eyaletin diğer eyalete girip çıkmaları vardır. Ve bunu Pong 9'da kullanacağız. Yani skoru göstermenin ötesinde, sadece başlangıç ​​ve oyun durumundan daha fazlasını takip etmeye başlayacağız. Aslında hizmet durumunu modellemeye başlayacağız. Öyleyse devam edip bunun neye benzediğini göstereyim. Eğer buradaysak Enter tuşuna bastım. Başlangıç ​​durumunda normal olarak başladık, ama Enter tuşuna bastım ve şimdi oyuncuya servis attığını söylüyor. Yani aslında hizmet ediyoruz. Bu yüzden, bana talimat verdiği gibi tekrar Enter tuşuna basarsam, birinci oyuncu solda, top sağa doğru hareket etmelidir. Hangi yapar. Bu yüzden devam edeceğim ve ikinci oyuncu olarak bilerek kaybedeceğim. Ve şimdi oyuncu iki servis attı. Yani hangi karakter olursa olsun, hangi oyuncu kaybederse tekrar servis atmalıdır. Ve şimdi Enter'a basarsam, birinci oyuncu olduğumuzu, top sağa hareket ettiğini not et. İkinci oyuncu için top sola doğru hareket eder. Şimdi biraz daha fazla etkileşimimiz var. Farklı eyaletlerimiz var. Oyuna başlıyoruz ve sonra servis yapıyoruz ve oynuyoruz. Yani top canlıyken, bunu gerçekten yaptığımızda, oyun halindeyiz. Şimdi hizmet durumundayız. Ne olmuş yani' Oyun durumu ile hizmet durumu arasındaki geçiş nedir? Oradaki geçiş nedir? Bir puan alıyoruz. Eğer durum diyagramımıza bakarsak ve oyun halindeysek, servis durumuna geçiş x oyuncunun puan kazanmasıdır. Ve sonra hizmet durumundaysak, geçiş birisi girmeye basar. Enter tuşuna basılır. Ve böylece, oyunumuzdan bir şekilde çıkıp kavramsal olarak düşünebileceğimiz bir dizi farklı mantık kümemiz varsa, oyunlarımız hakkında böyle düşünmek istiyoruz, bu da oyunumuzu farklı modları ve durumları bir araya getirebilir ve izlemesi gereken tüm bu değişkenler tarafından gerçekten fazla yükümlü olmaz - veya hangi durumda olduğumuzu? Tüm bu değişkenler ne yapıyor gibi? Ve biz' Önümüzdeki haftalarda bunu nasıl daha modüler bir şekilde çözebileceğimizi göreceğiz. Şu anda, şu anda yaptığımız tek şeyin, burada bazı dizelere bir durum değişkeni ayarlamak ve sadece küçük örnekler için fantastik bir şekilde çalışan koşullar varsa yapmak olduğunu unutmayın. Yani, örneğin, eğer güncelleme fonksiyonu ve oyun durumundaysak - bakınız, burada oyun durumunun sunulmak üzere ayarlanıp ayarlanmadığını söylüyoruz, o zaman tüm değişkenleri başlatıyoruz. Ve eğer oyun durumu oyun ise, o zaman mantığımızı burada gerçekleştirmeliyiz. Eğer eğer-- eğer oyundaysak, bu her kareyi arayacak ve eğer top bir oyuncu ile çarpışırsa, tüm bunları yapalım diyeceğiz. Ve sonra bu bizim oyunumuzu düşünmemize izin veriyor. Neredeyse güncelleme fonksiyonumuzda ayrı güncelleme fonksiyonlarına sahip olmak gibidir. Ve biz' Aslında, gerçek bir makine sınıfı ile gelecek haftalarda bunları bir güncelleme işlevinden nasıl çıkarabileceğimizi ve işleri biraz daha soyut olarak nasıl uygulayabileceğimizi göreceğiz. Ama şimdi, örneğin, bir geçiş yapmak istediğimizde, birisinin burada skor atması gibi, ekranın sol tarafına gidecekmiş gibi, top x sıfırdan daha az demektir. Tek yapmamız gereken bu durumu sunmak için ayarlamaktır ve güncelleme fonksiyonumuz daha sonra uygun şekilde güncellenecektir. Öyleyse burada Pong bağlamında devlet makinelerinin veya devletin nasıl çalıştığına dair sorularınız mı var? Evet. İZLEYİCİ: Yani devlet makinesi devletle ilişki mi yoksa devletlerin kabı mı? COLTON OGDEN: Devlet makinesi bir çeşit ... farklı durumlarınızın ne olduğuna ve geçişlerine genel bir kavramsal bakış, evet. Ve gelecek haftalarda biz burada bir durum makinesi nesnesi veya bir sınıf uygulamıyorsunuz. Ancak gelecek haftalarda, farklı durumlar arasındaki geçişleri daha modüler ve temiz bir şekilde yöneten devlet makine sınıfını göreceğiz. Burada yaptığımız tek şey devlet makinemiz, eğer ifadeler ve devletin buna eşit olup olmadığını söylemekse, o zaman bunu yapın. Ve sonra durumu bir değere değiştirin. İZLEYİCİ: Yani devlet makinesi bir kavram mı? COLTON OGDEN: Bir kavram. Evet. Ancak gelecek hafta bir devlet makinesinin uygulamasını bir nesne olarak göreceğiz. Başka soru? TAMAM. Güzel. Şu anda bir skorlama var. Gördüğümüz gibi, oyuncu bir skoru ve oyuncu iki skoru şimdi artmakta ve bu nedenle sola veya sağ kenara gittiğimizde ekrana getirilmektedir. Yani skoru takip ediyoruz. Ama birisinin kazanması için şimdi neye ihtiyacımız var? İZLEYİCİ: [duyulmaz]. COLTON OGDEN: Üzgünüm? İZLEYİCİ: [duyulmaz]. COLTON OGDEN: Evet öyle. Kesinlikle. Yani aslında oldukça basit. Gerçekten yapmamız gereken sadece bir if ifadesi, değil mi? Birinin puanı bir değere eşitse, 10, o zaman bazı oyuncular kazandı. Yani oyuncuya bakarsak - Pong 10'a bakarsanız, anaya gidip buraya, 174. ve 160. çizgiye gidiyoruz, kelimenin tam anlamıyla tüm mantığımızda olduğunu görüyoruz. Sadece topun sol veya sağ kenarın ötesine geçip geçmediğini görmek için test yapıyorsunuz. Çünkü bu, etkili bir şekilde birisinin bir puan kazanıp kazanmadığını görmek için çekinizi yapmanız gereken yerdir. Tek yaptığımız, programın bu bölümüne mantık eklemek ve puanlarını artırdıktan sonra söylemek. 10'a eşitse biz kazanan oyuncu olarak adlandırılan bir değer ayarlıyoruz. Bunu ikiye ayarlayacağız. Eğer ball.x sıfırdan küçükse, bu ekranın sol kenarından geçtiğinden oyuncunun attığı anlamına gelir. Bu nedenle, oyuncu iki puanı yükselmelidir. Ve bu nedenle, kazanan oyuncu da, eğer iki ya da iki oyuncu 10'a eşitse, ve bu durumda, buradayız, yeni bir durumu ayarlıyoruz ve sonra durum böyle değil veya eğer puanları hala 10'dan azsa, yine de servis için geri koymalı ve sonra topu sıfırlamalıyız. Ve eğer güncelleme fonksiyonumuza gidersek, burada-- aslında, bunu güncelleme safhamızda yapıyoruz. Yani, şu anda, eğer bitmiş durumdaysa, top sıfırlanır ancak bu durumda topa güncelleme uygulanmaz. Hâlâ puanlarımız var. 10. bu yüzden kimin 10 puanı varsa skoru diğer oyuncunun skorunu gösterir. Ve burada geçerli olan gerçek mantık, bu durumdan çıkmak istediğimizde, bizim aşkımızdadır. Tuşlu tuş işlevi. Oyun durumu 227. satırda olduğunuzu görüyoruz. Eğer oyun durumu daha önce ayarladığımız şuna eşitse ve bu sadece Enter veya return tuşuna basarsa çalışırveya Enter veya Return tuşuna basın. Oyunu kaydetmek için geri kaydetmek ayarlamak istiyorsunuz. Topu sıfırlamak istiyoruz. Bu puanları sıfırlamak istiyoruz. Bu yüzden etkili bir şekilde yepyeni bir oyun kuruyoruz. Kazanan oyuncu bir ise, o zaman bir sonraki oyunda avantaj elde edebilmeleri için ikramiye oyuncuya vereceğiz ve sonra bir tanesine ayarlayacağız. Ve render fonksiyonumuza inersek, 275 satırına inersek, eğer bitmiş durumdaysak, ekran oynatıcısına ve daha sonra kazanan oyuncuya render yapmalıyız, çünkü hatırlayın, kazanan oyuncuyu bir veya ikiye ayarladık , kimin kazandığına ve onuncu puanı kimin attığına bağlı olarak. Oyuncu bir veya iki kazanır diyecek ve bunu render edeceğiz ve daha sonra yeniden başlatmak için Enter'a basacağız. Ve bunun mantığı bu. Ve bunu playofflarda görebiliriz. Çok yavaşsa, bütün bir koşuyu geçmek zorunda kalmayabiliriz. Ama ben-- çemberler - topu gerçekten geri almak istiyorum. Hızı ayarladım ki bu-- servis durumundayız, oyun durumundayız, yükseldi - top geri döndü. Biraz sıkıcı olacak, ama söylemek yeterli, bu büyük bir kazanç. Endişelenme. Hızı biraz daha hızlı ayarlamalıydım. Neredeyse. Gerginleşiyor. Ve birinci oyuncu kazanır. İşte biz-- yazı tipini daha büyük bir boyuta ayarlıyoruz ve kodda temel olarak küçük yazı tipi ve büyük yazı tipi olan skor yazı tipi arasında yeni bir yazı tipi nesnesi oluşturuyorum. Yani 16 boyutlu yazı tipi. Ve böylece birinci oyuncu kazanır ve gerçekten de tüm bunlara dayanır. Sadece sayacınızı takip edin ve ekran çarpışmalarını tespit etmek için mantığınızda 10'a bastığınızda, devleti bitirdiğinizi Durum tamamlanırsa, klavye girişini izlemeniz ve ne zaman birisi girdiğinde görmeniz gerekir. Birisi aşkımızda Enter tuşuna basıyor. Tuşuna basıldığında, - oyuncu ayarı etkisi var - oyuncu iki servis çünkü bir oyuncu kazandı, bu sadece adil. Hizmet vermek için Enter tuşuna basıp yepyeni bir oyuna başlayacağız. Ve bu basit. Şimdi bir sürü basit durumla birlikte son derece oynanabilir bir oyunumuz var. Bence çok önemli bir ayrıntıyı kaçırıyoruz ve bu doğru. Şu anda, oyunumuz çok-- harika, oyun hepsi çalışıyor. Her şey iyi çalışıyor, ama sadece biraz cila eksik. Ve böylece yapacağımız şey oyuna ses eklemeye başlayacağız, bence, ekleyebileceğiniz daha eğlenceli şeylerden biri, çünkü aynı zamanda projenizin sonuna yaklaştınız. Love.audio.newsource burada bakacağımız bir işlevdir. Tüm bunlar bir yol ve sonra isteğe bağlı olarak bir tür almak olacaktır. Ve bu yol bir ses dosyasına gidecek ve uygulamanızın herhangi bir noktasında çalabileceğiniz bir ses nesnesi oluşturacak. Etkili bir şekilde yapacağımız şey, ne zaman bir çarpışma meydana geldiğinde, ne tür bir çarpışma olduğuna bağlı olarak, sadece belirli bir sesi çalacaktır. Ve tüm bunlar için gerçekten kullanmayı sevdiğim bir program, ve ben - projeniz için kendi seslerinizle uğraşmaya başlamak istiyorsanız sizi indirmenizi tavsiye ettiğim şey, bfxr adı verilen bir program. Windows ve Mac'te ücretsizdir. Linux portları olup olmadığından emin değilim. Bunun temelini oluşturan sfxr adlı benzer bir programın bir Linux portuna sahip olabilirler. Ancak bunun yapmanıza izin veren şey sadece bir grup rastgele ses üretmektir. Ve bunu kısaca sizin için açıklayabilirim. Eğer almak istiyorsanız, bfxr.net üzerindedir. Süper hızlı bir indirme ve-- burada, aslında-- göstereceğim, nasıl oynadığını görebilmemiz için. Yani arayüz bu. Ses bulduğumdan emin ol. Ve sonra burada birçok farklı ön ayar var. Yani pikap eğik çizgi, lazer eğik çizgi çekimi var. Bu tür küçük oyunlar gibi, anında prototip ses tipi şeyler uygulamak gibi. Ama görebilirsiniz, sadece uygular - Bunu geri çevireceğim, biraz gürültülü. Mesela, güçlendiricilerimiz var. Yani bunu her tıkladığımda, ' bize rastgele bir güç verecek. [BİLGİSAYAR SES ETKİLERİ] Ve sonra rastgele, her türlü garip kötü şeyler elde edersiniz. Ve sonra-- kullanacağımız şeyler blip slash select. Pong gibi arayüzler ve oyunlardaki şeylerin çoğu böyle basit sesler istiyor. Bu yüzden, oldukça iyi olduğunu düşündüğüm birkaç ses üretme işini zaten yaptım. Devam edip kodu önce Pong 11'de göstereceğim. Eğer - eğer görecekseniz - eğer dizin yapısına bakarsanız, bir sesler klasörümüz olduğunu göreceksiniz. Sesler klasöründe, kürek topa vurduğunda üç ses, kürek vuruşu oluşturdum. Herhangi bir zamanda topun ekranın sol veya sağ sınırını aştığı zaman skor. Ve sonra duvara çarptı, böylece top ekranın üstüne veya altına dokunduğunda. Ve böylece bunun mantığı son derece basittir. Yapmamız gereken tek şey ne zaman-- zaten uyguladık, yani tek yapmamız gereken-- ah, ilk şey, söylemeliyim, ve bu bir tablonun iyi bir örneğidir. Ve bunu gelecekte çok daha fazla görmeye başlayacağız. Bu derste tabloları çok fazla kullanmadık, ama tablo Lua'nın hepsi gibi, hepsi de veri yapısı gibi. Bu sözlük-- Python sözlüğü, JavaScript nesnesi. Bu bir dizi, Lua'daki basit değişkenlerin ötesinde bir şey için ihtiyacınız olan her şey. Diğer kütüphanelerdeki sınıflar gibi bile her şey yapılır. Bu durumda, burada ses adı verilen bir tablo başlatıyoruz ve üç anahtardan geçiyoruz, bu yüzden alır - anahtar değer çiftlerini alabilir, ya da sadece bir değerler listesi verebilir ve bunlar dolaylı olarak onlar için endeksler yaratacaktır. Burada, sadece Python veya JavaScript'te yaptığınız gibi geçiyoruz. Raket vurmak ve burada bu biçimde gibi bir tablodaki anahtar / değer çiftlerini başlatmak için bu köşeli parantezlere ihtiyaç duyduğunu unutmayın. Raket hit aşk alır. Ses.newsource. Ve bu durumda, sadece bir yol alır, bu yüzden sesler kürek başı nokta dalgasını keser. Ve biz ona anahtar kelime veriyoruz - ya da statik, yani varlık türü - dize olarak depolanıyor. Böylece statik veya akışlı ses öğeleriniz olabilir. Yani statik iseler, belleğe yüklenir ve programınızın yürütülmesi için bellekte tutulurlar. Eğer akışsa, oyun motorunuzun gerektirdiği şekilde anında yüklenir. Akan ses varlıkları, tonlarca ses ve büyük ses dosyaları gibi uzun bir oyununuz varsa yardımcı olabilir. Bunların hepsini hafızada tutmak istemezsiniz, çünkü bu çok, çok, çok sayıda meg veya ses işleyebilir. Varlıkları anında yüklüyorsanız, oyununuzda dinamik yüklemeniz varsa, bu dikkate almanız gereken başka bir şeydir. Bu durumda, bunlar çok küçük ses dosyalarıdır, çünkü sanki bir saniye sürenin bir kısmı gibidirler. Bu yüzden hepsini statik olarak ayarlıyoruz, böylece bellekte korunuyorlar ve sadece üçünü de bu tabloya yüklüyoruz. Ve daha sonra bunlara değinmek istiyorsak, tek yapmamız gereken sesler - ya seslere referans verebiliriz. Varsayılan olarak, Lua size sadece bir nokta anahtar kelimesi verir, JavaScript'in nesnelerini, geçirdiğiniz anahtarla aynı ada sahip bir şekilde yapma veya Pythonic yolunu yapabilirsiniz, nokta olmadan, üzgünüm - sadece köşeli parantezlerle ve şimdi aynı - bu ikisi eşdeğerdir. Yine de işe yaramaz, eğer anahtarınıza bir boşluk koymaya karar verirseniz, inanacağım, işe yaramaz - hiç işe yaramaz, ancak bir alt çizgi enjekte edebilir. Test edip bulmam gerekecek. Ancak, genel olarak, zaten bu şekilde sıralanmış tuşlara sahip olduğunuzda nokta kullanmak en iyi yöntem değildir. Ve nokta gösterimini kullanarak nokta ile yapamayacağınız dizelerle yapabileceğiniz şey, dinamik olarak sizinle tablonuzun bir görünümünü oluşturmaktır - ki bu da yapabilirsiniz - evet, çünkü yapamazsınız dört döngüde tabloda dört şey yapın ve sonra bir şeyleri noktalayın, bu işe yaramaz. Ancak tablonuzdaki her şey için yapabilir ve daha sonra anahtarı bu değer, tablonuzdaki yinelenen değer olarak arayabilirsiniz. Gelecek derslerde bunun örneklerini göreceğiz. Ama bu sadece akılda tutulması gereken bir şey. Burada masamız var, bir ses tablosu. Oh, ve-- ve seslerimiz hazır, hafızaya yüklenmişler. Şimdi tek yapmamız gereken, kodumuzdaki herhangi bir çarpışma gibi, sadece bunu yaptığımız her şeye sahip olduğumuz yer. Masa kadar basit. İstediğimiz anahtarda, Lua'nın bir sınıfın veya masanın işlevini çağırmanın yolu olan iki nokta üst üste. İki nokta üst üste çalma ve çalma işlevi, tabloda oluşturduğumuz AŞK'daki yeni kaynak ses nesnesinin bir parçasıdır ve sadece bir kez çalma etkisi olacaktır. Döngüye ayarlayabilirsiniz. Kürek isabetini söyleyebilir, döngüyü true olarak ayarlayabilir, ve sonsuz bir şekilde tekrar tekrar çalacak, ki böyle bir ses istemezdik. Kulağa iğrenç gelebilir. Ancak, örneğin bir seviyede ya da bunun gibi bir müzik parçanız varsa, döngü sona erdiğinde doğru olmasını istersiniz, böylece sonunda sona erdiğinde, kullanıcınız sadece sessizce bir oyun oynamaz. Bu yüzden raket vuruşu ile yapıyoruz, duvar vuruşu ile de yapıyoruz. Ses dosyalarının nerede ve hangi amaçla kullanıldığını bulmak kolay olur. Bu nedenle, üst veya alt sınırlarda olduklarında, duvara çarpma sesini çalın ve ardından top ekranın soluna veya sağ kenarına her ulaştığında, puan ses efektini çalın. Ve eğer oyunumuzu oynarsak, ve bu her zaman en sevdiğim parçalardan biri oyunu sesli olarak oynamaktır, çünkü bence böyle bir fark yaratıyor. Ses efektleri alıyoruz. Bu küçük bir şey ve çok kolay, ama ekliyor - çok fazla lezzet katıyor. Ve sonra (patlama), işte başlıyoruz. Ve sonra oyunumuz bu noktada pratik olarak uygulanmaktadır. Size göstermek istediğim bir örnek daha var, küçük bir örnek, çünkü şu ana kadarki tüm örnekler yeniden boyutlandırma eşittir - yeniden boyutlandırılabilir eşittir yanlış. Push kurulum ekranı başlatıcısında bir çeşit anahtar ve pencerenizi yeniden boyutlandırabileceğiniz bir oyuna sahip olmak istiyorsanız, tek yapmamız gereken, genişlik ve yükseklik alan love.resize adlı bir işlevi çağırmaktır. Ve bununla, özellikle de kullanım durumumuz için bunu yapacağız, çünkü itme kullanıyoruz, Pong 12'ye gideceğiz. Ve sonra main.lua'ya gidersek, burada 85 satırında görüyoruz, aslında uygulamayı yeniden boyutlandırmamıza izin verecek şekilde şimdi yeniden boyutlandırılabilir olarak eşit olanı değiştirdim. Bu yanlışsa, ekranın alt köşesini tıklayıp sürükleyemezsiniz, sadece yapmanıza izin vermez. Ve sonra yapmanız gereken tek şey aşkı aramaktır. Yüksekliği yeniden boyutlandırın ve sonra yükseklikle birlikte itme boyutuna geçin. Çünkü kaputun altına itmek bir doku alır ve ona yansıtır ve ardından pencerenizi doldurmak için ölçeklendirir ve böylece doğru pencere boyutlarına uyacak şekilde yükseltmek için mevcut pencere boyutlarınızın ne olduğunu bilmesi gerekir. Ayrıca push, aynı en boy oranını korumak istiyorsanız kullanışlı olan mektup kutusu gibi şeyler de ekler. Ve belki de uygulamanızın büyüklüğüne göre oluşturulan kullanıcı arayüzüne sahip olduğunuz bir oyunda, bu işlev önemli olacaktır çünkü o zaman yeniden boyutlandırabilirsiniz - UI öğelerinizi uygun şekilde yeniden boyutlandırabilir ve yeniden konumlandırabilirsiniz. Çünkü oyununuz küçükse, belki bir ton ekran alanı kaplamamak ve sadece uygulamanızın tüm olası kullanıcılarını barındırmak için kullanıcı arayüzünün belirli bölümlerinin görünmez olmasını veya tamamen farklı bir konumda olmasını istersiniz. . Ama bunun şu etkisi var - eğer Pong 12'ye gidip çalıştırırsak, aslında, bunu bile kullanamayabiliriz ama evet, bunu şimdi yapabilirim. Yeniden boyutlandırabilirim ve daha önce ayarladığımız sanal genişlik ve yüksekliği koruyacak, çünkü her şeyden önce itmenin ne yapacağına benziyor ve uygulamanızın boyutu ne olursa olsun uygulamanızın boyutu ne olursa olsun mektup kutusu olacak bu en boy oranı. Yani eğer ' dikey veya yatay olarak bu en / boy oranının ötesinde, bunun için uygun posta kutusunu alırsınız. Bu yüzden süper uygun. Kullanıcılarınızın süper çarpık en-boy oranları elde etmeleri konusunda endişelenmenize gerek yok, çünkü öngörülemeyen bir çeşit çözünürlük kullanıyorlar. Süper küçük olsa bile her zaman koruyacaktır çünkü monitörleri süper incedir. Her zaman en boy oranını koruyacaktır. Ama aslında Pong için bu kadar. Tam bir oyunumuz var, bitirmeye başlıyoruz ve herhangi bir sorunuz varsa, onlara cevap vermekten mutluluk duyarız. Sorusu olan? Güzel. Tamam. Bu kursun geri kalanını size öğretmekten heyecan duyuyorum çocuklar. Sadece yüzeyi çizdik. Ele alacağımız çok şey var. Gelecek hafta, aslında Flappy Bird'ü ele alacağız, bu yüzden güzel renkli grafikler alacağız, Bugün siyah beyaz estetiğimiz için önemli bir fark. Ama Pong için bu kadar. Geldiğiniz için teşekkürler.