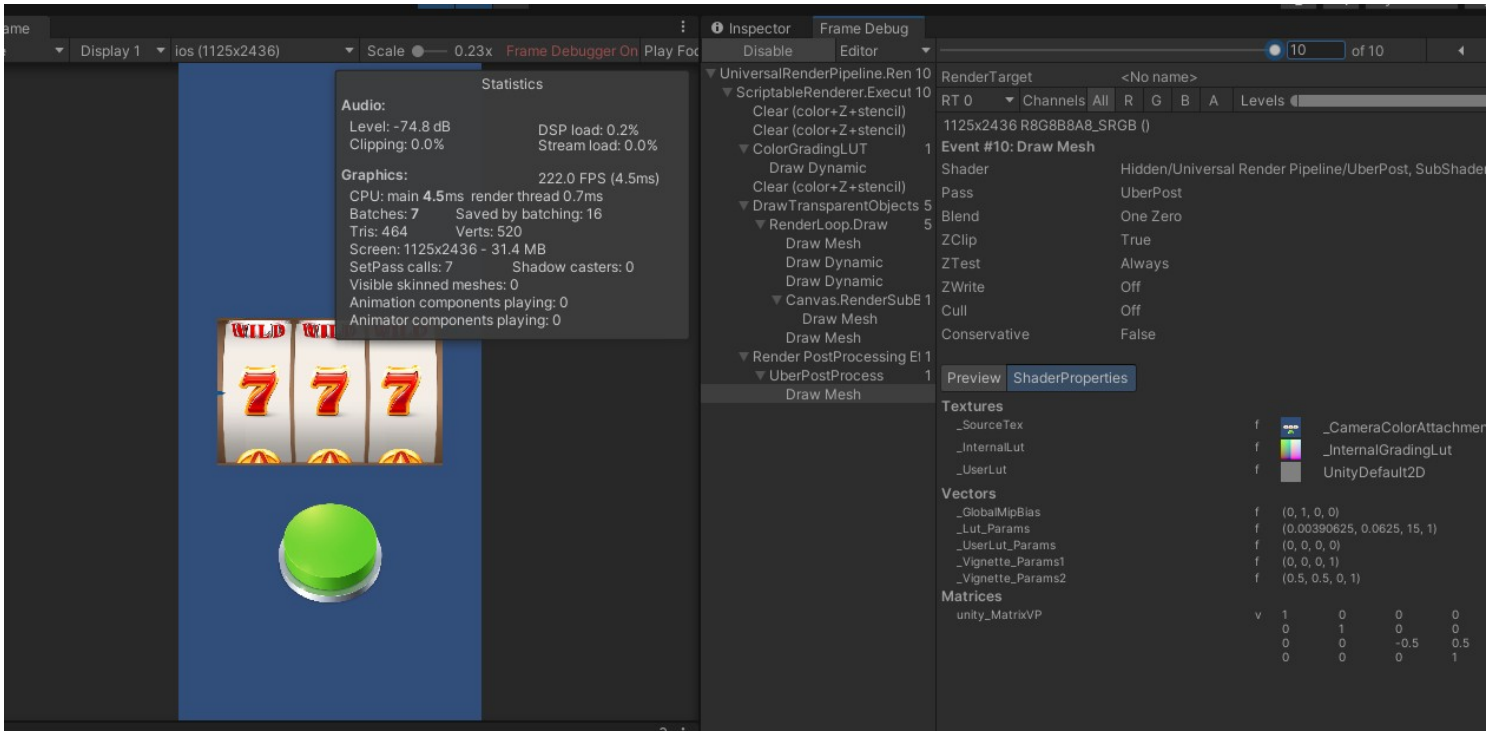
Selamlar Fuat, öncelikle detaylı geri bildirimin için tekrardan çok teşekkürler. Ben de bu detaylı incelemene karşı mümkün olduğunca istenilenleri yapıp projeyi güncelledim. Anlatması biraz uzun olacağından ve görseller de ekleyeceğimden bir bütün olarak pdf halinde göndermek daha mantıklı gözüktü. Adım adım yazdığın maddelere değinecek olursak:

- Oyundaki scroll mekaniği için ne yapacağımdan tam emin değildim aslında. Sprite renderer ve mask kullanımı daha mantıklı gelmişti hatta öyle de başladım. İlk başlarda o efekti scroll rect kullanıp vertical ui group kullanaraktan daha kolaylıkla sağlayabilir miyim derken UI sistemi olarak bıraktım en son. Onun haricinde UI kullanmamın arkasında bir motivasyon yok dediğin gibi performans açısından çok daha kötü aslında. Belirttiğin gibi Uı sistemini tamamen kaldırıp oyunu tamamen GameObject üzerinden ilerleyecek şekilde çevirdim.

- Oyun daha ufak çapta olduğu için dikkat etmemiştim ama evet haklısın, bu kadar ikonun render edildiği bir yerde SpriteAtlas kullanmak daha mantıklı. Bu eklemeyi yapınca da aşağıdaki değerleri aldım zaten.



- Yine dediğin gibi tam olarak emin olmak için unit testleri kullanmak en sağlıklısı. Oyunun logic kısmını scriptable object ve monobehaviorlardan (kısaca Unity'den) ayırıp hem unit test yazmamı sağlayacak bir yapı kurdum ve hem de görevleri dağıttım. Yine ilerleyen maddelerde bahsedicem bundan.

- SpinGenerator ın hem datayı provide etme, hem logic işleme hem de cacheleme görevleri vardı. SpinDataHolder adındaki classa data provide etme kısmını ekledim, datayı cachelemeyi ise SaveHandler a aktardım. Bu sayede SpinGenerator ona verilen datayı içindeki logic ile işleyen bağımsız bir class haline geldi.

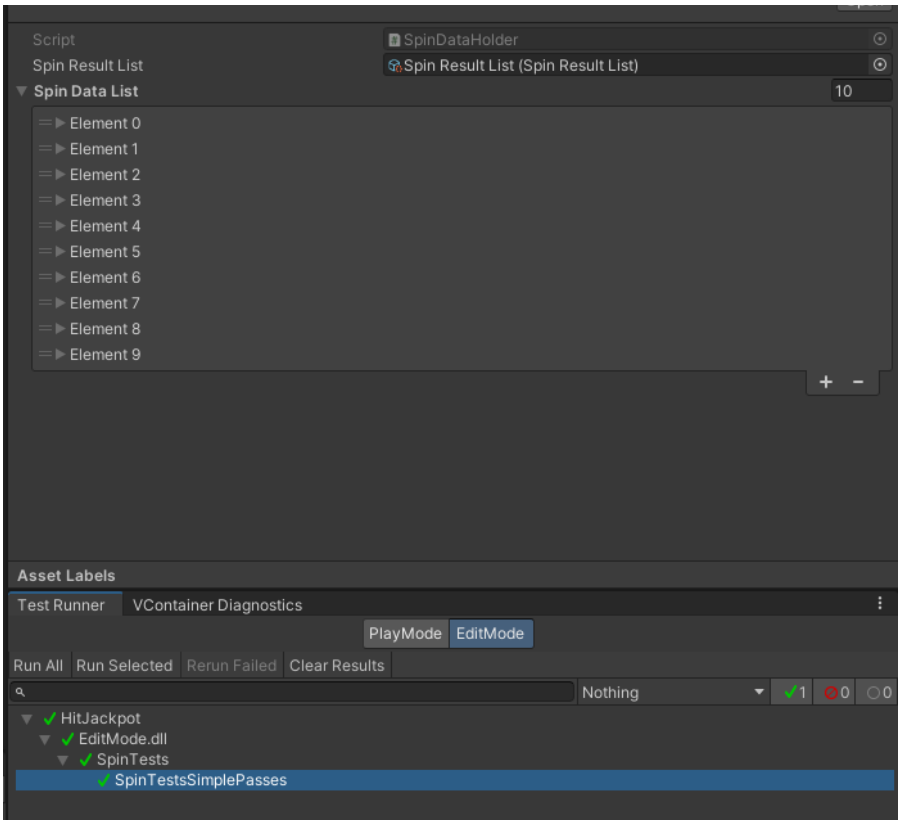
- SpinResult structunun içine aldım söylediğin IsFull değişikliğini :)

- Core ve GameElements dediğin gibi draftta kalmış bir namespace sistemiydi. Unit test eklemediğim için çok detaylandırma gereği duymamıştım ama şimdi hem test eklendiği, hem de daha fazla sınıf oluşturulup kendi sorumluluklarına ayrıldığı ve Dependency Injection eklendiği(bir sonraki maddede detaylandırdım) için şu anda çok daha detaylı bir namespace sistemi var klasörlemeye bağlı olaraktan.

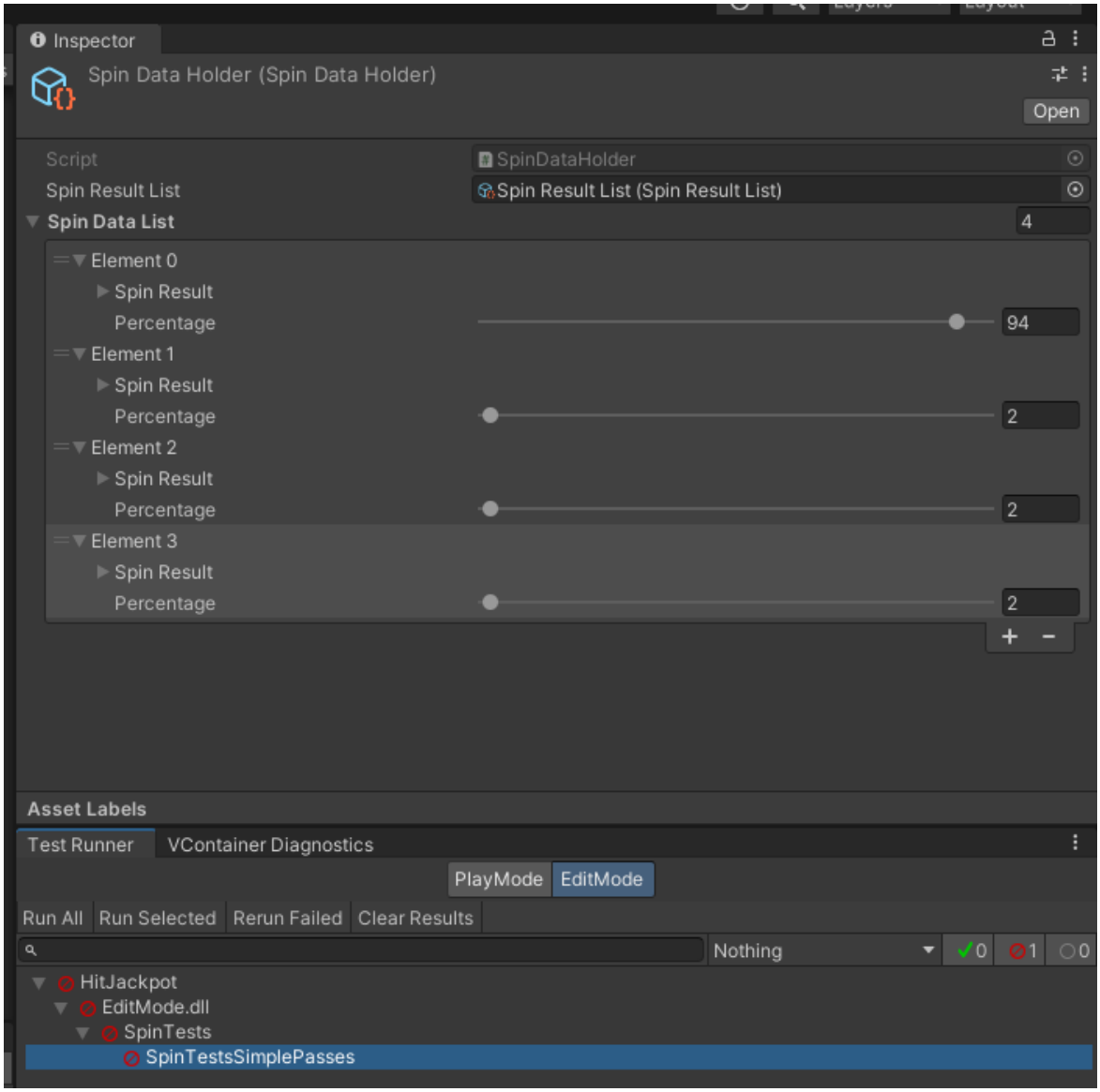
- En önemlisi büyük ihtimalle, unit test için data providerları (ağırlıklı olarak scriptable object) ve monobehaviorlar olarak tanımladıktan sonra, logic kısımları için ayrı, unityden bağımsız classlar oluşturdum.(SpinGenerator, SlotController, SaveSystemHandler,SpinButtonSystem) Normalde data provide işlemi için, tüm datalar ScriptableObjectlerde, sahneden bağımsız bir şekilde tutarak yapıyordum. Ama bu dataya erişmesi gereken birden fazla logic sistemi için (eski haliyle monobehaviorlar, şu anki haliyle logic classları) sahnedeki her MonoBehaviour için ScriptableObject de olsalar referanslarını tek tek girmem gerekiyordu. Bu logic sistemleri MB' lerden ayrılınca doğal olaraktan datayı provide etmek daha da zor ve dolaylı oluyordu. Bu yüzden bu gereksinimi karşılaması için Dependency Injection kullandım. Benim bildiğim Unity için bir tek zenject ile vcontainer var, vcontaineri daha detaylı denemek istiyordum bahane oldu diyebilirim :) Yine zenject ile veya genel olarak bir DI frameworkü ile aynı mantıktan, bir tane DIContainer tüm ilgili dependent componentleri register edip, herhangi bir register olmuş class constructurinda çağrılan parametreleri provide ediyor. Bu sayede hem sahnedeki referans karmaşasından kurtuldum, hem de aynı şekilde editor unit test yapabilecek şekilde bir logic classı yazabildim.

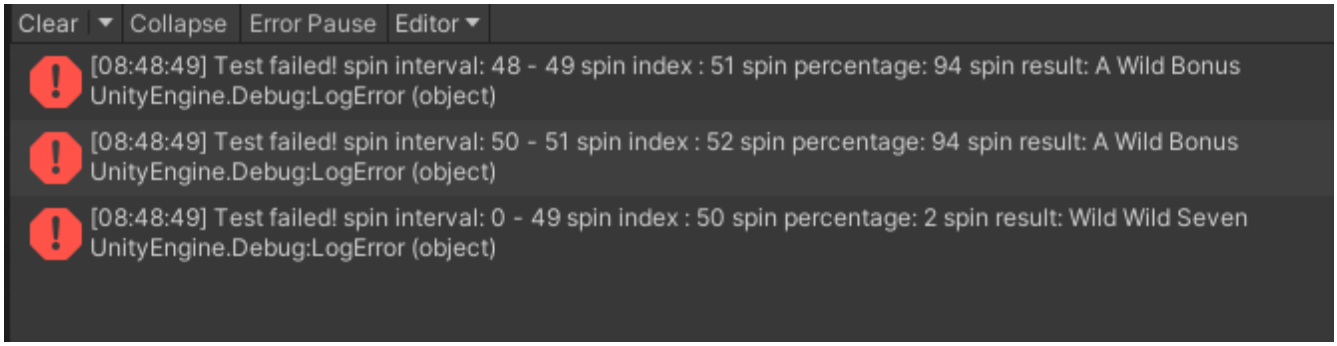
- "_ =" kullanımında aslında ilk iki spin her zaman üçüncüden önce biteceğinden onları bekleme gereği duymadım. Aynı şekilde belirttiğin gibi compiler warningleri engellemesi açısından koymuştum. Dönüş sırasından bağımsız hala gelmesi için Task.WhenAll yapısına çevirdim.

- Bunun haricinde, bahsi geçen olayları yorumda anlattım ama en gözlemlenebilir olanı unit testle desteklenen hali oluyor. Aşağıdaki unit testlerde, her bir spinin olması gereken aralık ve kaçıncı spinde geldiğine bakılaraktan bir unit test yapılıyor. Eğer tüm gelen sonuç indexleri, olması gereken kendi aralığındaysa test başarılı, onun haricinde hatalıysa hatalı olan tüm indexleri ve aslında olması gereken değer aralığını bize her sonuç için yansıtıyor.



Buradaki casede verilen olasılıklarla yaplılan test, ekstradan bir görsel feedback vermedim ama yukarıda anlattığım mantıkla çalışan bir test olduğundan sonuçların doğruluğu kesin. Onun haricinde yorumlarda da bahsettiğim stacking olayına örnek vermek gerekirse:





Bahsettiğim ekstrem değerlerden, 94 – 2 – 2- 2 olasılıkları verilince test hata veriyor, hataya baktığımızda ise 100 spinden 3 ünün hatalı olduğunu görüyoruz. Normalde tam bölenlerin arasına girmesi gerekiyordu, ama algoritma en erken gelmesi gereken (intervali bitmesine en yakın ve hala gelmemiş) resultu tercih ettiği için, 94 yüzdesi zaten bir overhead yaratıyor. Onun harici kalan küçük olasılıkların hepsi aynı olunca, hepsinin gelmesi gereken an aynı anda geldiği için, olasılıklardan biri kendi aralığını bir aksatabiliyor. (Örn. [50-51] aralığında gelmesi gerekirken 52 gelmesi). Yüzde yüz mükemmel dağılım olmasa da sanırım casede bahsedilen mükemmele yakın dağılıma uyuyor. Çünkü casedeki örneği incelediğimde

Wild, Bonus, A (%13)

0-7	8-15	16-23	24-31	32-39	40-47	48-54	55-61	62-69	70-77	78-85	86-93	94-99	
7	14	21	28	35	42	49	56	63	70	78	86	94	OK
1	10	17	30	38	40	49	56	68	75	80	88	99	OK
4		20,22	30		41,43	49	57	65	72	80	88	99	X

Normalde yüzde 13 lük bir olasılığın mükemmel dağılımında, 9 tane 8 aralıkta ve 4 tane 7 aralıkta bir gelmesi gerekir. (100/13 = 7 ve 100 % 13 = 9, mükemmel dağılım için 9 kalanı 9 aralığa bir bir dağıtılması) Ama örnekte de baktığımızda aslında bu dağılımın aksine, 10 tane 8 aralıkta, 2 tane 7 ve bir tane de 6 aralık (sonuncusu aralık) var ve bu onaylanan bir durum olarak alınmış. Mükemmele yakından kasıt bu sanırsam, bu denkleme karşılık arada bir iki indexlik sapma olabiliyor, veya ben biraz fazla kurcaladım olayı bilmiyorum :D

Son olarak da belirttiğin gibi GenerateSpinListNew biraz haddinden büyük bir fonksiyon olmuştu, onu refaktörledim.

Ayırdığın vakit ve yorumların için tekrardan teşekkürler.