International Journal of Innovative Engineering Applications

Journal homepage: https://dergipark.org.tr/ijiea

A HYBRID ANDROID FRONTEND DEVELOPMENT APPROACH WITH TRADITIONAL NESTING LAYOUTS AND CONSTRAINT LAYOUT

Yusuf Özçevik*[1]

[1]*Department of Software Engineering,Manisa Celal Bayar University, Manisa,Turkey*

Abstract

Original scientific paper

Android operating system is a mobile platform that draws attention with the variety of devices it supports and its widespread use. On the other hand, devices with different sizes and screen resolutions require application developers to consider multi-screen support. For this, different layout objects in the Android Software Development Kit (Android SDK) are used by dividing them into parts in certain proportions. While using traditional nested layouts in old Android SDK versions; it is recommended to use constraint layout in the recent versions. On the other hand, converting structures in existing projects to entirely constraint layouts creates both a maintenance cost for the project lifecycle and a learning cost for the developers. Accordingly, this study proposes a hybrid multi-screen support approach using traditional nested layouts and the constraint layout, together. The performance of the proposed approach is compared with the screen loading times of the traditional method and the contemporary method. As a result, it is observed that there is no significant performance difference, especially on new generation devices with up-to-date hardware; whereas it is revealed that devices with old generation hardware experience performance problems when the number of nested layouts used increases. Thus, appropriate use cases for the proposed hybrid approach are discussed.

Keywords: *Android programming, multi-screen support, traditional nested layouts, constraint layout.*

GELENEKSEL İÇ İÇE YERLEŞİMLER VE KISIT YERLEŞİMİ İÇEREN MELEZ ANDROİD ÖN YÜZ GELİŞTİRME YAKLAŞIMI

Özet

Orijinal bilimsel makale

Android işletim sistemi, desteklediği cihaz çeşitliliği ve yaygın kullanım oranı ile dikkat çeken bir mobil platformdur. Öte yandan, farklı boyut ve ekran çözünürlüğüne sahip cihazlar, uygulama geliştiricilerin çoklu ekran desteğini dikkate almalarını gerekli kılar. Bunun için, Android Yazılım Geliştirme Kiti (Android YGK) içerisinde bulunan farklı ön yüz nesneleri belirli oranlarda parçalara bölünerek kullanılmaktadır. Eski Android YGK sürümlerinde geleneksel iç içe yerleşimler kullanılırken; güncel sürümlerde ise kısıt yerleşimi kullanımı tavsiye edilmektedir. Öte yandan, mevcut projelerdeki yapıların tamamen kısıt yerleşimi içeren ön yüzlere dönüştürülmesi, hem proje yaşam döngüsü için bakım maliyeti hem de geliştiriciler için öğrenme maliyeti oluşturmaktadır. Buna göre, bu çalışmada, geleneksel iç içe yerleşimler ile kısıt yerleşiminin kullanıldığı melez bir çoklu ekran desteği yaklaşımı önerilmektedir. Önerilen yaklaşımın performansı, geleneksel yöntemin ve güncel yöntemin ekran yükleme süreleri ile kıyaslanmaktadır. Sonuç olarak, özellikle güncel donanıma sahip yeni nesil cihazlarda kayda değer bir performans farklılığı yaşanmadığı görülürken; kullanılan iç içe yerleşimlerin sayısı arttıkça, eski nesil donanıma sahip cihazlarda performans sorunları yaşandığı ortaya konmaktadır. Böylece, önerilen melez yaklaşım için uygun kullanım durumları tartışılmaktadır.

Anahtar Kelimeler: *Android programlama, çoklu ekran desteği, geleneksel iç içe yerleşimler, kısıt yerleşimi.*

1 Giriş

Modern bilişim teknolojilerindeki gelişmeler geçtiğimiz yüzyılda hızlanarak günümüzde kullanılan bilgisayar destekli sistemlerin temelini oluşturmuştur [1]. Yirminci yüzyılın ortalarında tanıtılan ilk genel kullanım amaçlı bilgisayardan bu yana donanım bileşenlerinin boyutu hızla küçülürken, sahip oldukları işlem yeteneği son derece artmıştır [2]. Donanım teknolojilerindeki bu gelişmeler kişisel bilgisayar kavramını ortaya çıkararak önce sabit konumlu daha sonra taşınabilir bilgisayarların yaygınlaşmasına neden olmuştur. Günümüzde ise taşınabilir bilgisayar donanımları daha da küçülerek cepte taşınabilir, giyilebilir hatta insan vücuduna yerleştirilebilir

* Corresponding author.

E-mail address: yusuf.ozcevik@cbu.edu.tr (Y. Özçevik)

Received 13 May 2022; Received in revised form 14 September 2022; Accepted 15 November 2022

2587-1943 | © 2022 IJIEA. All rights reserved.

Doi: https://doi.org/10.46460/ijiea.1116222

hale gelmiştir [3]. Bilgisayar donanımları ile ilgili yaşanan teknolojik gelişmeler bilgisayar yazılımlarının hedef platformunu, teknik özelliklerini ve mimari yapısını büyük oranda değiştirmiştir. Başlangıçta hantal ve büyük iş istasyonları için geliştirilen yazılımlar genelde bir konsol ekranı üzerinden son kullanıcı etkileşimi sağlarken, kişisel bilgisayarların hayatımıza girmesiyle masaüstü yazılım uygulamaları yaygınlaşmıştır. Diğer yandan, iletişim teknolojilerindeki gelişmeler ve yirminci yüzyılın sonlarına doğru kişisel kullanım için yaygınlaşan internet kavramı ile yazılım uygulamalarının tarihsel seyri tamamen değişmiştir [4]. İnternet kavramının günlük hayatta kullanılmasıyla birlikte son kullanıcı uygulamaları büyük oranda web platformlarına kaymıştır. Ayrıca, iletişim teknolojilerinin yarattığı bu değişim donanım teknolojileri için yeni hedefler belirlemiştir. Yirmi birinci yüzyılın başlarında akıllı telefonların ve mobil işletim sistemlerinin tanıtılmasıyla yazılım uygulamaları için yeni bir hedef platform ortaya çıkmıştır [5]. Dahası, mobil işletim sistemlerinin yaygınlaşmasıyla, bilgisayar sistemlerinde süregelen donanımsal değişim akıllı telefonlarla sınırlı kalmayıp; giyilebilir donanımlar, otomobiller, akıllı ev aletleri gibi günlük hayatta insan etkileşimine konu olan birçok nesneye yansımıştır.

1.1 Android İşletim Sistemi ve Android Uygulamaları

Android işletim sistemi, farklı tür donanımlara yazılım geliştirmek için kullanılan en yaygın platformlardan biridir [6]. 2008 yılında yayımlanan ilk kararlı sürümüyle birlikte başlangıçta akıllı telefonlara yönelik uygulama geliştirmeye olanak sağlasa da, ilerleyen yıllarda desteklediği donanım çeşitliliği artmıştır [7]. Akıllı telefonlar için günümüzde sağlıktan, eğitime; eğlenceden, finans sektörüne kadar birçok farklı alanda Android uygulamaları mevcuttur. Akkara vd. göz doktorları için klinik veri toplamak ve hasta takibi yapmak üzere kullanılan Android telefon uygulamalarını inceleyerek farklı teknoloji seviyesinde yönlerini ortaya koymaktadır [8]. Venkataraman vd. öğretmen eğitimi müfredatının yürütülmesi ve takip edilmesi için eğitim alanında bir Android cep telefonu uygulaması geliştirmektedir [9]. Sweidan vd. eğitim ve oyun kavramlarını birleştiren sanal gerçeklik tabanlı bir telefon uygulaması önermektedir [10]. Dieter vd. bankacılık uygulamaları için geliştirilen mobil uygulamaların özelliklerine ilişkin bir çalışma ortaya koymaktadır ve güvenlik tehditlerinin nasıl önleneceğine vurgu yapmaktadır [11].

Telefon uygulamalarının yanı sıra, Android işletim sistemine sahip akıllı tabletler için geliştirilen birçok mobil uygulama mevcuttur. Matic vd. akıllı tablet ile kalem kullanmadan, sadece insan parmağı ile çizim yapmak üzere bir çizim uygulaması geliştirmiştir [12]. Papadakis vd. çocuklar için geliştirilen eğitici tablet uygulamalarının sosyal, duygusal ve bilişsel gelişime uygunluğunu araştırmaktadır [13]. Maher vd. bir Android tablet uygulamasıyla ormanların gerçek zamanlı izlenmesi ve orman sağlığı için aykırı durumların yönetilmesine ilişkin bir uygulama önermektedir [14].

Donanım sektöründeki gelişmelere paralel olarak, 2014 yılından bu yana Android işletim sistemine sahip televizyonlar için uygulama geliştirme imkânı ortaya çıkmıştır. Bu yönden, farklı amaçlarla akıllı televizyon uygulamaları geliştirmek mümkündür. Aafer vd. akıllı televizyonlarda yer alan uygulamalar için anomali tespiti yaparak güvenlik açıklarını ortaya çıkarmaya yarayan bir akıllı televizyon uygulaması önermektedir [15]. Skorupska vd. ise, TEDx çeviri sisteminde yer alan hataların tespitini kitle kaynak kullanımı ile sağlamak amacıyla, özellikle yaşlı bireylerin kullanımını hedefleyen, bir akıllı televizyon uygulaması ortaya koymaktadır [16].

2015 yılından bu yana ise, giyilebilir cihazlar ve otomobiller için uygulama geliştirmek üzere Android uygulama geliştirme araçları yaygın olarak kullanılmaktadır. Jisha vd. tarafından geliştirilen uygulamada, çocukların ebeveynler tarafından uzaktan takip edilmesi için, okul servisinin ve öğrencilerin konumu bir saat ile izlenmektedir [17]. Pajic vd. çalışmalarında, Android uygulamalar ile bilgi ve eğlence tabanlı yeni nesil otomobil tasarımının mümkün olduğunu göstermektedir [18]. Ayıca, Android işletim sistemi ile geliştirilen ve farklı uygulama alanları için dağıtılan uygulamalar, tüm bu donanımlarla sınırlı kalmayarak nesnelerin interneti kavramına yönelik gömülü yazılım ürünlerini de içermektedir [19]. Dahası, Android uygulamalar her çeşit donanıma ayrı ayrı geliştirilebileceği gibi, bulut tabanlı bir ekosistemde farklı istemciler üzerinde koşmak üzere geliştirilebilir [20].

1.2 Çoklu Ekran Desteği Problemi

Android işletim sistemi koşan ve Android uygulama çalıştırmayı destekleyen donanımların bu denli çeşitliliğe sahip olması, yazılım geliştiriciler için bazı zorluklar ortaya çıkarmıştır. Bu zorlukların en başında çoklu ekran desteği gelmektedir. Çoklu ekran desteği bir Android uygulamasının, farklı çözünürlük ve boyuta sahip cihazlarda, aynı kullanıcı ön yüzü ile gösterilerek aynı kullanıcı deneyimini yansıtması anlamına gelmektedir [21]. Profesyonel bir yazılım ürününün, farklı boyut ve çözünürlükte ekranlar içeren her donanımda uygun şekilde görüntülenmesi gerekir. Apple şirketi tarafından sunulan iOS yazılımları sadece Apple tarafından desteklenen standart bir donanım kümesi üzerinde çalışırken; açık kaynak kodlu Android işletim sistemi çok sayıda dağıtıcı tarafından kullanılmaktadır ve geniş bir donanım kümesinde yer alan cihazlarca desteklenmektedir.

Android platformunda çoklu ekran desteği probleminin üstesinden gelmek için literatürde farklı yaklaşımlar yer almaktadır. Bunlardan biri çapraz platform uygulama geliştirme olarak bilinen yaklaşımdır. Bu yaklaşım kullanılarak, tek bir kaynak kodu tabanı ile farklı platform ve donanımlara uygulama çözümleri geliştirmek mümkündür. Biørn-Hansen vd. çalışmalarında, çapraz platform uygulama geliştirmenin performansını incelemiştir [22]. Yazarlar, bu yaklaşımın performans olarak tatmin edici uygulamalar geliştirmeye olanak sağladığını belirtmiştir. Diğer yandan, çapraz platform uygulama geliştirme kütüphanelerinin kullanıcı deneyimi konusunda aynı başarıya sahip olup olmadığı tartışılabilir. Google, Android platformu için tasarlanan uygulamalarda *Material Design* adı verilen yerel tasarım kalıplarını önermektedir [23]. Benzer şekilde, Apple, iOS

uygulamalarda kullanıcı deneyimini belirleyen *Cupertino* tasarım kalıplarını ön plana çıkarmaktadır [24]. Yerel kütüphanelerde yer alan bu tasarım kalıplarının, en yüksek kullanıcı deneyimi için oluşturulduğu göz önüne alındığında; çapraz platform uygulama geliştirme yaklaşımlarının, yerel uygulama geliştirme yaklaşımları kadar kullanıcı deneyimi sunamadıkları bilinmektedir. Android uygulamaların çoklu ekran desteği sorununa çözüm olarak uygulanan bir diğer yaklaşım, profesyonel yazılım uygulamalarının sadece belirli cihazlar tarafından çalıştırılmasını koşullandırmaktır. Diğer bir deyişle, uygulama, geliştirilen tasarımın uygun şekilde görüntülendiği cihazlara yüklenebilirken; diğer cihazlara yüklenmesi engellenmektedir. Guan vd. çalışmalarında, ekran kısıtlarıyla ilgili koşullandırmaların *Manisfest.xml* dosyasında gerçekleştirilebileceğine değinmektedir [25]. Oysaki, bu durum, çoklu ekran desteği problemini tam anlamıyla çözmek yerine, uygulamanın dağıtılabildiği donanım sayısını kısıtlayarak, sorunu yokmuş gibi göstermektedir. Daha uygun bir yaklaşım ise, uygulamanın hedeflediği tüm cihazların ekran özellikleri göz önüne alınarak, ön yüz tasarımının her bir cihaz için en yüksek kullanıcı deneyimi ile son kullanıcılara sunulmasıdır. Bu yaklaşım, zaman ve kaynak maliyetinin yönetimi açısından uygulanabilir görülmese de otomatik ön yüz oluşturan bazı araçlar ile daha verimli hale getirilebilir. Chen vd. farklı ekran boyut ve çözünürlüğüne sahip donanımlar için otomatik ön yüz oluşturan bir araç tanıtmaktadır [26]. Ne var ki, bu tür araçların lisanslı kullanımı bir ücretlendirmeye tabi olup, üstüne öğrenme maliyeti de eklenmektedir. Android çoklu ekran desteği için en yaygın kullanılan yaklaşımın, oransal bir ekran tasarımı ile sağlandığı söylenebilir. Bu yaklaşımda, geleneksel sırasal yerleşim ve bağıl yerleşim nesneleri ile uygulama ekranı belirli oranlara bölünerek, her bir yerleşim içerisine boyutları cihaz ekranına oransal olarak sınırlandırılmış bir ön yüz nesnesi eklenmektedir. Bu durum, iç içe yerleşim sayısı arttıkça bileşen ağacı derinliğini artmasına sebep olmaktadır ve Android işletim sistemi tarafından ön yüzün ekrana yüklenme süresini artırmaktadır. Mondal vd. birçok donanımda karşılaşılan ön yüzün yavaş oluşturulma probleminin sebebi olarak çok sayıda iç içe sırasal ve bağıl yerleşimler kullanılmasını göstermektedir [27]. İç içe yerleşimler sebebiyle uygulama ön yüzlerinin ekrana yavaş yüklenmesi problemi, 2018 yılında tanıtılan AndroidX sürümündeki kısıt yerleşimi ile ortadan kaldırılmıştır [28]. Kısıt yerleşimi, bileşen ağacının aynı düzeyinde yer alan bileşenler arasında bazı nitelikler tanımlayarak oransal çalışmaya olanak sağlamaktadır. Böylece, iç içe yerleşimler kullanıldığında ortaya çıkan bileşen ağacı derinliğinin artması durumu önlenmektedir. Bielik vd. kısıt yerleşimleri ile hızlı, güvenli ve çoklu ekran desteği sağlayan ön yüzler tasarlanabileceğini belirtmektedir [29]. Kısıt yerleşimleri, çoklu ekran desteği sorununa büyük oranda çözüm oluştururken; öğrenme maliyeti ve mevcut projeler için ön yüzlerin kısıt yerleşimi kullanarak tekrar oluşturulması sorununa yol açmaktadır.

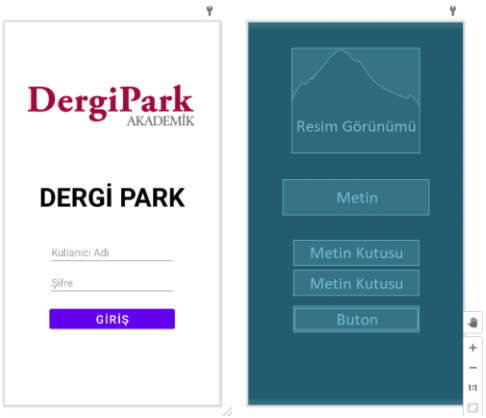
Yukarıda belirtilen motivasyonlara göre, Android işletim sisteminin yaygın kullanımı ve çoklu ekran desteğine ilişkin zorluklar dikkate alındığında bu çalışmada önerilen fayda aşağıdaki gibi sıralanabilir:

- Çoklu ekran desteği için geleneksel iç içe yerleşimler ve kısıt yerleşimi içeren melez bir ön yüz geliştirme yaklaşımı önerilmektedir.
- Önerilen melez yaklaşımın performansı geleneksel iç içe yerleşimlere göre ve kısıt yerleşimine göre ayrı ayrı kıyaslanmaktadır.
- Farklı cihaz donanımları ve farklı sayıda bileşen içeren ön yüz nesnelerinden oluşan bir değerlendirme ortamı hazırlanarak değerlendirme sonuçları ortaya konmaktadır.

Çalışmanın geri kalanında ilk olarak Android ön yüz geliştirme yaklaşımları ve karşılaşılan performans probleminin açık tanımına yer verilmektedir. Sonrasında önerilen yaklaşımın detayları ve performans değerlendirmesine ilişkin bulgular ortaya konarak çalışma sonlandırılmaktadır.

2 Android Ön Yüz Geliştirme Yaklaşımları

Android uygulama geliştirirken tasarlanan ön yüzler için farklı yaklaşımlar kullanılabilir. Bu durumu somutlaştırmak için Şekil 1 ile gösterilen örnek bir uygulama ön yüzü tasarımı tanıtılmaktadır. Şekilde yer alan tasarımın kullanıcı girişi sağlayan bir Android ekranı için geliştirildiği düşünülebilir. Bu tasarımın en üst kısmında logo içeren bir resim görünümü, onun altında başlık bilgisi içeren bir metin, orta kısımda kullanıcı adı ve şifre bilgilerini almak için iki adet metin kutusu ve en alt kısımda giriş yapmak için bir buton yer almaktadır. Şeklin sol tarafında yer alan tasarım penceresi ile sağ tarafında yer alan ve ön yüz bileşenlerinin konumlanmasını gösteren pencere Android Studio tümleşik geliştirme ortamından (TGO) bir ekran görüntüsü ile elde edilmiştir. Bu tasarımın geliştirilmesi için sadece kısıt yerleşimi kullanabileceği gibi, geleneksel yerleşimler içeren iç içe bileşenler de kullanılabilir. Kullanılan yaklaşım, bileşen ağacı derinliğini değiştireceği için uygulama performansını etkilemektedir.

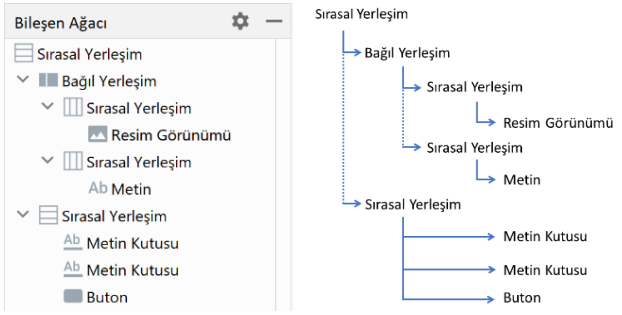


Şekil 1: Bir Android uygulaması için hazırlanan örnek ön yüz tasarımına ilişkin Android Studio ortamından alınan ekran görüntüsü.

2.1 Geleneksel Yerleşimler

Geleneksel yerleşimler, Android YGK'nın erken sürümlerinden itibaren kullanılan yerleşim bileşenlerinden oluşmaktadır. Sırasal yerleşim ve bağıl yerleşim yaygın kullanılan geleneksel yerleşimlerden ikisidir. Sırasal yerleşim, içerisinde bulunan bileşenleri

yatayda veya dikeyde sıralayarak bir görünüm oluşturur. Bağıl yerleşim ise, içerdiği bileşenleri kendisine göre veya birbirlerine göre bağıl olarak konumlandırmak için kullanılmaktadır. Bu iki yerleşim bileşenini kullanarak herhangi bir ön yüz tasarımını geliştirmek mümkündür. Çoklu ekran desteğine yönelik olarak oransal çalışmak için sırasal yerleşimlerin ve bağıl yerleşimlerin iç içe kullanılması gerekebilir.

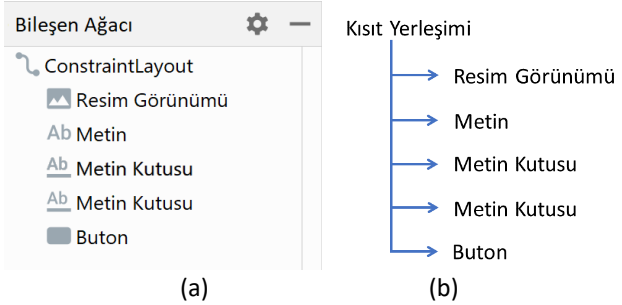


Şekil 2: Geleneksel iç içe yerleşimler ile oluşturulan bir ekran tasarımının bileşen ağacına ilişkin (a) Android Studio geliştirme ortamından alınan ekran görüntüsü (b) bileşenlerin mantıksal dizilimi.

Şekil 1 ile verilen tasarımı geleneksel iç içe yerleşimler kullanarak gerçekleştiğimizi göz önüne alırsak, Şekil 2 ile gösterilen bir bileşen ağacı elde edilebilir. Şekil 2.a Android Studio TGO üzerinden alınan ekran görüntüsünü yansıtırken; Şekil 2.b bu ön yüzde bulunan bileşenlerin mantıksal hiyerarşisini göstermektedir.

2.2 Kısıt Yerleşimler

AndroidX sürümü ile birlikte geleneksel yerleşimlere alternatif olarak kısıt yerleşimi kavramı ortaya çıkmıştır. Buna göre, iç içe yerleşimlere gerek kalmadan, bileşenleri bazı kısıtlar tanımlamak suretiyle konumlandırmak mümkündür. Kısıt yerleşimleri için, çoklu ekran desteği, bileşenlerin en ve boy niteliklerine oranlar atayarak elde edilmektedir. Şekil 1 ile verilen ön yüz tasarımı, geleneksel iç içe yerleşimler yerine sadece kısıt yerleşimi kullanarak geliştirilirse, Şekil 3 ile gösterilen bir bileşen ağacı elde edilebilir.



Şekil 3: Kısıt yerleşimi ile oluşturulan bir ekran tasarımının bileşen ağacına ilişkin (a) Android Studio geliştirme ortamından alınan ekran görüntüsü (b) bileşenlerin mantıksal dizilimi.

Şekil 3.a Android Studio TGO üzerinden alınan ekran görüntüsünü yansıtırken; Şekil 3.b bu gerçeklemenin mantıksal hiyerarşisini göstermektedir. Buna göre, kısıt yerleşimi kullanıldığında elde edilen bileşen ağacı, geleneksel yerleşimler kullanıldığında elde edilen ağaca göre daha az bileşene sahiptir ve bileşen ağacı derinliği daha azdır. Ayrıca, yerleşim dosyalarında kullanılan içerik sayısı azaldığı için kaynak kodun boyutu da azalmaktadır.

Kısıt yerleşiminin geleneksel yerleşimlere göre avantajlı görünmesinin yanı sıra, özellikle karmaşık ön yüz tasarımlarının kısıt yerleşimleri kullanılarak dönüştürülmesi maliyetli bir operasyondur. Bu maliyet, mevcut projeler için hem zaman maliyetini hem de geliştirici ekip için öğrenme maliyetini içermektedir. Öte yandan, Android TGO tarafından sunulan otomatik bazı ön yüz dönüştürücüleri mevcuttur. Bu araçlar kullanılarak, geleneksel yerleşimlerle hazırlanmış bir ön yüz, kısıt yerleşimine dönüştürülebilir. Ancak, çoklu ekran desteğinin kapsamı göz önüne alındığında, kısıt yerleşimi kullanarak, aynı tasarımı farklı bileşen ağaçlarıyla ifade etmek mümkündür. Bu nedenle, bu tür dönüşüm araçları, bileşenlerin yerleşimini olası dönüşümlerden birini seçerek gerçekleştirir. Sonuçta, dönüştürülen kodların mutlaka tekrar incelenmesi ve gerekirse düzenlenmesi gerekir.

2.3 Problem Tanımı

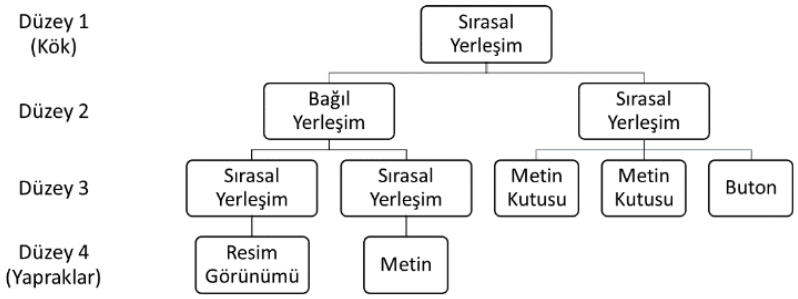
Android işletim sistemi, bir ön yüzde bulunan bileşenleri ekranda göstermek üzere ölçüm, yerleşim ve çizim adında üç temel adım çalıştırmaktadır. Bu adımlarda ele alınan işlemler aşağıdaki gibi tanımlanmaktadır:

Ölçüm: Her bir bileşenin ve içerdiği alt bileşenlerin ölçülerini belirlemek için bileşen ağacı kökten yapraklara doğru taranır ve ölçüler hesaplanarak kaydedilir.

Yerleşim: Bileşen ağacında kökten yapraklara doğru bir tarama daha gerçekleşir ve her bir ön yüz bileşeni için ölçüm aşamasında hesaplanan boyutlar kullanılarak nesnelerin ve içerdikleri alt nesnelerin konumları belirlenir.

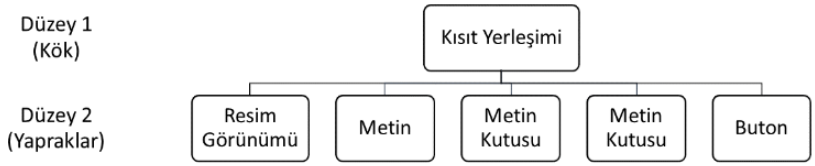
Çizim: İşletim sistemi bileşen ağacı üzerinde kökten yapraklara doğru bir tarama daha yapar. Bileşen ağacındaki her nesne için, GPU'ya bir çizim komutları listesi göndermek üzere bir Canvas nesnesi oluşturulur. Bu komutlar, sistemin önceki 2 aşamada belirlediği bileşenlerin boyutlarını ve konumlarını içerir. Böylece, bu adımın sonunda, tasarımda yer alan her bir bileşen ekranda belirir.

İşletim sistemi ön yüz nesnelerini ekrana yüklerken, yukarıda belirtilen her bir adım için bileşen ağacını tekrar taradığı için, bileşen ağacı derinliği uygulama performansını etkilemektedir. Bu durumu, çalışmada ele aldığımız örnek üzerinden somutlaştırmak gerekirse, Şekil 4 ile verilen bileşen ağacı kökten yapraklara doğru uzanan ön yüz düğümlerinden oluşmaktadır. Kök düğümü ekranı saran bir sırasal yerleşim nesnesidir. Yapraklar, ekranda görüntülenen resim görünümü nesnesi, metin nesnesi, metin kutusu nesnesi (2 adet) ve buton nesnesinden oluşmaktadır. Arada yer alan diğer düğümler ise çoklu ekran desteği için oranlamaların yönetildiği yerleşim nesnelerini temsil etmektedir. Ele alınan bu yapıda, bileşen ağacının derinliği 4 olarak hesaplanabilir. Bu ön yüzde yer alan her bir bileşen çizim adımları sırasında işletim sistemi tarafından üçer kez taranmaktadır.



Şekil 4: Geleneksel yerleşimler ile oluşturulan örnek ön yüzün ekranda görüntülenmesi sırasında işletim sistemi tarafından taranan bileşen ağacı.

Çalışmada ele alınan örnek ön yüz tasarımı sadece kısıt yerleşimi kullanılarak çizdirildiğinde, bileşen ağacı Şekil 5 ile gösterilen yapıda olmaktadır. Kısıt yerleşimi ile tasarlanan ön yüzde, çoklu ekran desteği için fazladan yerleşim nesnelerine gerek duyulmamaktadır. Bu nedenle, kök düğümü doğrudan yapraklarla ilişkilendirilebilir. Oransal düzenlerin oluşturulması içinse kısıt tanımlamalarından faydalanılır. Sonuç olarak, ağacın derinliği 2 olacağı için, içerisinde yer alan tüm bileşenlerin işletim sistemi tarafından ekrana yüklenmesi sırasında ağaç taraması sebebiyle meydana gelen zaman kaybı azalacaktır.

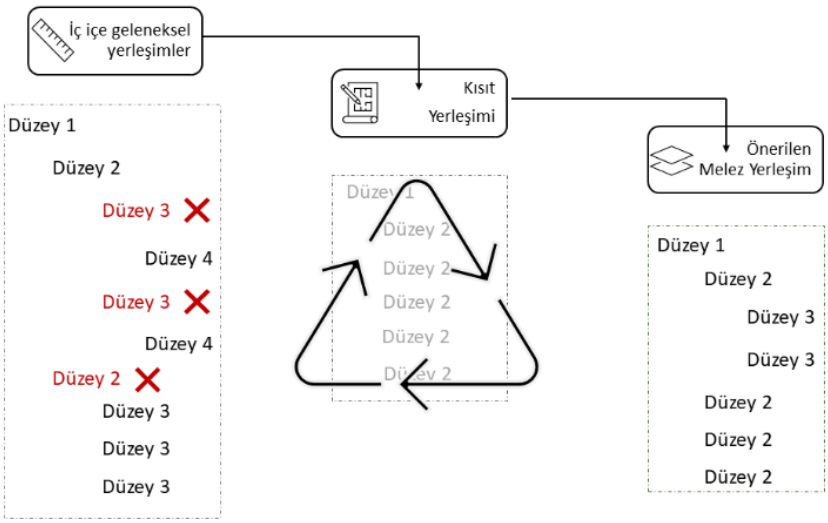


Şekil 5: Kısıt yerleşimi ile oluşturulan örnek ön yüzün ekranda görüntülenmesi sırasında işletim sistemi tarafından taranan bileşen ağacı.

Kısıt yerleşiminin geleneksel yerleşimlere göre performans üstünlüğü, ön yüz bileşenlerinin işletim sistemini tarafından ekrana çizdirilme süreciyle doğrudan ilişkilidir. Bu nedenle, bileşen ağacı derinliğinin belirli düzeyin üstüne çıkması, uygulamalarda ön yüzün yavaş yüklenmesi sorununu doğurmaktadır. Bu durum, özellikle düşük işlem yeteneğine sahip cihazlarda, kullanıcı deneyimi için bazı olumsuzlar ortaya çıkarmaktadır. Öte yandan, mevcut projeler için kısıt yerleşimi uygulanırken ortaya çıkan öğrenme maliyeti olabildiğince düşük düzeyde tutulmalıdır. Böylece, uygulama ile sağlanan kullanıcı deneyimi ve geliştirme maliyeti arasında uygun bir denge kurulabilir.

3 Önerilen Yaklaşım

Bu çalışmada, geleneksel iç içe yerleşimler ile modern kısıt yerleşimini birlikte kullanarak her iki yaklaşımın avantajlarına sahip melez bir yaklaşım önerilmektedir. Bu sayede, kısıt yerleşimlerinin uygulanması sırasında karşılaşılan maliyetlerin azaltılması ve geleneksel yerleşimlerin performans sorununun aşılması hedeflenmektedir. Bu amaçla, ele alınan yaklaşımın farklı karmaşıklıktaki ön yüz tasarımlarına uygulanması ve farklı donanımlarda ne gibi performans farklılıklarının ortaya çıktığını gözlemlemek üzere Şekil 6'daki blok diyagram ile verilen süreç takip edilmektedir.



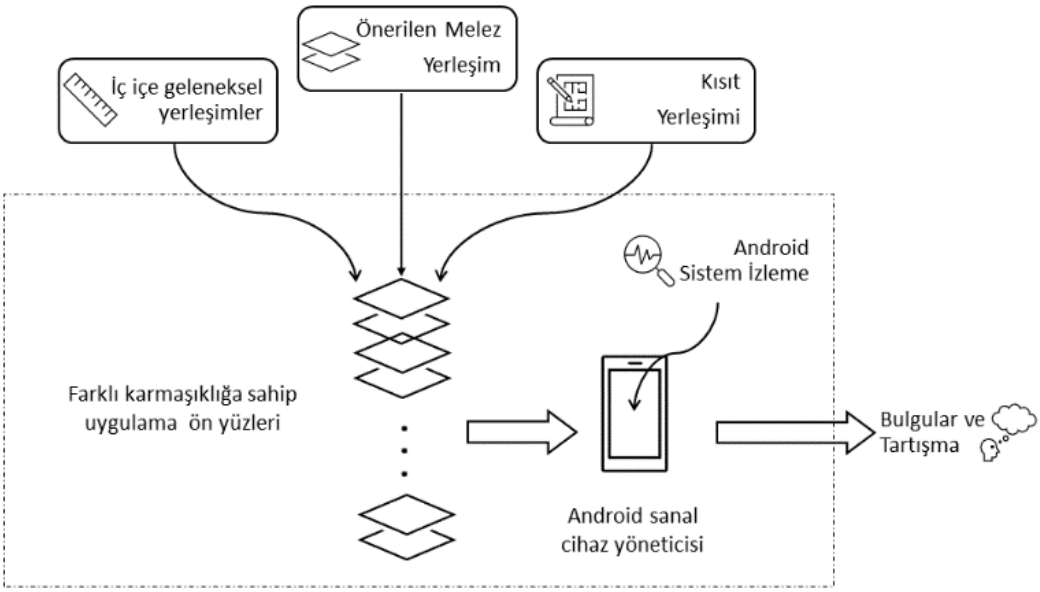
Şekil 6: Önerilen melez ön yüz geliştirme yaklaşımının uygulanmasına ilişkin blok diyagram.

Şekilde yer alan blok diyagram için önceki bölümlerde tanıtılan örnek ekran tasarımı ele alınmaktadır. Buna göre, ilgili ekran tasarımında sadece geleneksel iç içe yerleşimler kullanmak yerine; geleneksel yerleşimlerin kısıt yerleşimi ile birlikte kullanılması önerilmektedir. Böylece, önerilen melez ön yüz geliştirme yaklaşımında oluşan bileşen ağacının derinliğinin azaltılması hedeflenmektedir. Bu durumu Şekil 6 üzerinde somutlaştırmak gerekirse, sadece geleneksel yöntem ile elde edilen ve iç içe bileşenlerden oluşan soldaki ön yüz tasarımında yer alan yapraklar en fazla dördüncü düzeyde yer almaktadır. Diğer bir deyişle, ekrana çizdirilen ön yüz bileşenlerinin tutulduğu bileşen ağacı derinliği dörttür. Bu yapıda yapraklardan hemen önce yer alan ve kırmızı renk ile belirtilen *Düzey* 3 ve *Düzey 2* bileşenleri, kısıt yerleşiminin tasarıma dahil edilmesiyle birlikte bileşen ağacından kaldırılabilir. Bu sayede, en sağda gösterilen ve melez yaklaşım ile elde edilen bir ön yüz tasarımında yaprakların sahip olduğu en yüksek düzey üçüncü düzey olarak belirlenmektedir. Böylece, bileşen ağacının derinliği dörtten üçe düşürülerek azaltılmaktadır.

Önerilen melez yaklaşımda, geleneksel iç içe yerleşimlerin çoklu ekran desteği için oransal çalışma özelliği ve kullanım kolaylığından faydalanılır. Diğer yandan, kısıt yerleşimlerinin farklı geleneksel yerleşimler içerisinde parçalı olarak kullanılmasıyla performans kazancı elde edilmeye çalışılır. Sonuç olarak, hem geleneksel yerleşimlerin hem de kısıt yerleşiminin avantajlı yanlarını ortaya çıkaran melez bir ön yüz geliştirme yaklaşımı elde edilir.

4 Performans Değerlendirmesi

Çalışmada önerilen melez ön yüz geliştirme yaklaşımı ile, geleneksel iç içe yerleşimleri ve kısıt yerleşimini birlikte kullanarak elde edilen ön yüz tasarımlarının performans kıyaslaması ele alınmaktadır. Performans ölçümleri için hazırlanan değerlendirme ortamı ve içerdiği bileşenler Şekil 7 ile gösterilmektedir. Buna göre, her bir yöntemle farklı sayıda bileşen içeren ve farklı bileşen ağaçları ile temsil edilen ön yüz tasarımları oluşturularak bir dizi deney tekrarlanmaktadır. Her bir deneyde, çalışmada ele alınan üç yaklaşım ile ayrı ayrı tasarlanan ön yüzlerin ekrana yüklenme süreleri ölçülmektedir. Böylece, farklı ön yüz karmaşıklarına sahip ön yüz tasarımları için hangi yöntemin daha verimli çalıştığı araştırılmaktadır.



Şekil 7: Performans değerlendirmesi için hazırlanan değerlendirme ortamı ve bileşenleri.

Performans değerlendirmesi için hazırlanan deneyler Android sanal cihaz yöneticisi tarafından yönetilen farklı donanım performansına sahip 3 adet cihaz üzerinde koşmaktadır. Böylece, kullanılan yaklaşım ve ön yüz tasarım karmaşıklığının farklı donanımlar üzerinde nasıl bir performans değişimi gösterdiğinin araştırılması hedeflenmektedir. Performans değişiminin incelenmesi için Android TGO'da yer alan sistem izleme özelliğinden faydalanılmaktadır. Android sistem izleme, bir uygulamanın bir cihaza yüklenmesiyle birlikte bir izleme oturumu başlatmaktadır. Bu izleme oturumunda, uygulamanın cihaz üzerinde nasıl bir performans sergilediğini gözlemlemek üzere bellek kullanımı, çalışma süresi, enerji tüketimi gibi farklı performans kriterleri takip edilebilmektedir. Ayrıca, uygulamanın çalışması sırasında görev alan her bir proses ve iplik detaylıca incelenebilmektedir. Bu nedenle, çalışmada bir tasarımın yüklenmesi sürecini yöneten iplikler takip edilmektedir. Android işletim siteminde yer alan *UIThread* ve *RenderThread* iplikleri ön yüz tasarımlarının ekrana yüklenmesine ilişkin operasyondan sorumludur. Sistem izlemesi sırasında bu ipliklerin çalışma süreleri takip edilerek her bir tasarımın ekrana ne kadar sürede yüklendiği ayrı ayrı kaydedilmektedir. Sonuçta, farklı donanımlar üzerinde farklı tasarımlar için elde edilen bulgular birbirleriyle karşılaştırılarak ortaya konmakta ve tartışılmaktadır.

4.1 Değerlendirme Ortamı

Değerlendirme ortamı için 3 farklı cihaz donanımı ve 8 farklı ön yüz tasarımı ele alınmaktadır. Farklı ön yüzlerin her biri, çalışmada incelenen yaklaşımlara göre ayrı ayrı tasarlanarak her bir cihazda koşturulmaktadır.

Performans deneylerinin ele alındığı ortam 16 GB RAM, 3.70 GHz işlemci hızı ve 64-bit adresleme yeteneği olan Windows 10 işletim sistemine sahip bir bilgisayar ortamıdır. Ön yüz tasarımlarının geliştirilmesi ve performans incelemesi için Android Studio TGO kullanılmıştır. Android sanal cihaz yöneticisi tarafından oluşturulan 3 farklı cihazın teknik özellikleri Tablo 1 ile gösterilmektedir.

Tablo 1: Değerlendirme ortamı için ekran çizim sürelerinin hesaplandığı farklı Android cihazların teknik özellikleri.

Cihaz donanımı	CPU sayısı	RAM miktarı	Heap boyutu
Eski nesil	2	1 GB	128 MB
Standart	4	4 GB	256 MB
Yeni nesil	8	8 GB	512 MB

Tablo 1'e göre, çalışmada kullanılan cihazlar eski nesil, standart ve yeni nesil olmak üzere üç farklı işlem yeteneği ile ele alınmaktadır. Eski nesil cihaz donanımda 2 CPU, 1 GB RAM ve 128 MB Heap boyutu bulunmaktadır. Standart donanıma sahip cihazda 4 CPU, 4 GB RAM ve 256 MB Heap boyutu bulunurken; yeni nesil donanıma sahip cihaz ise 8 CPU, 8 GB RAM ve 512 MB Heap boyutu içermektedir. Farklı yaklaşımlar ile tasarlanan ön yüzlerin ekrana yüklenme süreleri her bir cihaz üzerinde ayrı ayrı ölçülerek elde edilen bulgular paylaşılmaktadır.

Performans değerlendirmesi için farklı sayıda bileşen içeren farklı ön yüz tasarımları ile 8 farklı deney tekrarlanmaktadır. Bu ön yüz tasarımlarının her biri 3 farklı yöntem ile ayrı ayrı geliştirilerek 3 farklı cihaz donanımı için ekrana yüklenme süreleri incelenmektedir. İç içe geleneksel yerleşimler kullanıldığında elde edilen bileşen ağacı özellikleri Tablo 2 ile belirtilmektedir.

Tablo 2: Değerlendirme ortamında ele alınan farklı karmaşıklığa sahip ön yüz tasarımları için geleneksel iç içe yerleşimler kullanıldığında oluşan bileşen ağacı özellikleri.

	Yaprak sayısı	Düğüm Sayısı	Ağaç Derinliği
Deney 1	4	10	5
Deney 2	6	20	7
Deney 3	9	30	8
Deney 4	11	40	10
Deney 5	13	50	11
Deney 6	14	60	13
Deney 7	16	70	14
Deney 8	19	80	15

Tablo 2'ye göre, her bir deney için bileşen ağacındaki yaprak sayısı, ön yüzde görüntülenen nesne sayısına karşılık gelmektedir. Düğüm sayısı, iç içe yerleşimler oluştururken kullanılan ara düğümleri de içeren toplam bileşen ağacı düğümlerini gösterirken, ağaç derinliği ise ilgili bileşen ağacındaki en yüksek düzey değerine sahip nesnenin düzeyini göstermektedir.

Kısıt yerleşimi kullanıldığında elde edilen bileşen ağacı özellikleri Tablo 3 ile belirtilmektedir. Her bir deney için, farklı yaklaşımlarla aynı ön yüz tasarımı ele alındığından, yaprak sayıları Tablo 2'de belirtilen değerler ile aynıdır. Diğer yandan, kısıt yerleşimi tasarımında yer alan her bir ön yüz nesnesi birbirlerine göre veya ekrana göre kısıtlar kullanılarak konumlandırılmaktadır. Bu nedenle, kısıt yerleşim nesnesi hariç, her bir nesnenin düzey sayısı ve dolayısıyla ağaç derinliği 2 değerini almaktadır. Toplam düğüm sayısı ise yaprak sayısının bir fazlasına eşittir.

Tablo 3: Değerlendirme ortamında ele alınan farklı karmaşıklığa sahip ön yüz tasarımları için kısıt yerleşimi kullanıldığında oluşan bileşen ağacı özellikleri.

	Yaprak sayısı	Düğüm Sayısı	Ağaç Derinliği
Deney 1	4	5	2
Deney 2	6	7	2
Deney 3	9	10	2
Deney 4	11	12	2
Deney 5	13	14	2
Deney 6	14	15	2
Deney 7	16	17	2
Deney 8	19	20	2

Kısıt yerleşimi ve geleneksel iç içe yerleşimleri birlikte kullanan melez yaklaşıma ilişkin her bir deneyde elde edilen bileşen ağacı özellikleri Tablo 4 ile verilmektedir. Buna göre, yaprak sayıları diğer iki yöntemde karşılık gelen her bir deneydeki yaprak sayıları ile aynı iken; düğüm sayısı ve ağaç derinliği değerleri, diğer iki yöntemde yer alan değerlerin arasında değerler almaktadır. Böylece, önerilen melez yaklaşım ile geleneksel yerleşim tasarımında yer alan ara düğümlerin bir kısmı kaldırılırken, bileşen ağacının derinliği azaltılmaktadır.

Tablo 4: Değerlendirme ortamında ele alınan farklı karmaşıklığa sahip ön yüz tasarımları için önerilen melez yaklaşım kullanıldığında oluşan bileşen ağacı özellikleri.

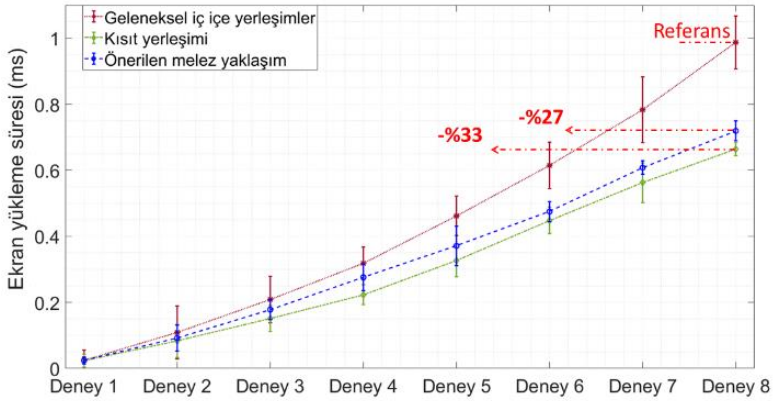
	Yaprak sayısı	Düğüm Sayısı	Ağaç Derinliği
Deney 1	4	8	3
Deney 2	6	14	4
Deney 3	9	18	5
Deney 4	11	26	5
Deney 5	13	33	6
Deney 6	14	41	6
Deney 7	16	49	7
Deney 8	19	58	7

Yukarıda tanıtılan değerlendirme ortamı dikkate alındığında 8 farklı ön yüz tasarımı, 3 farklı yaklaşım ile hazırlanarak, ekrana yüklenme süreleri 3 farklı cihaz donanımında ayrı ayrı ölçülmüş ve elde edilen değerlendirme bulguları paylaşılmıştır. Ölçülen süreler cihazın o anki toplam işlem yüküne bağlı olduğundan, her bir durum için deneyler 100 kez tekrarlanmaktadır. Tekrarlar sonucu elde edilen ekrana yüklenme sürelerinin ortalaması alınarak, incelenen yöntemler birbirleriyle kıyaslanmaktadır.

4.2 Elde Edilen Bulgular ve Tartışma

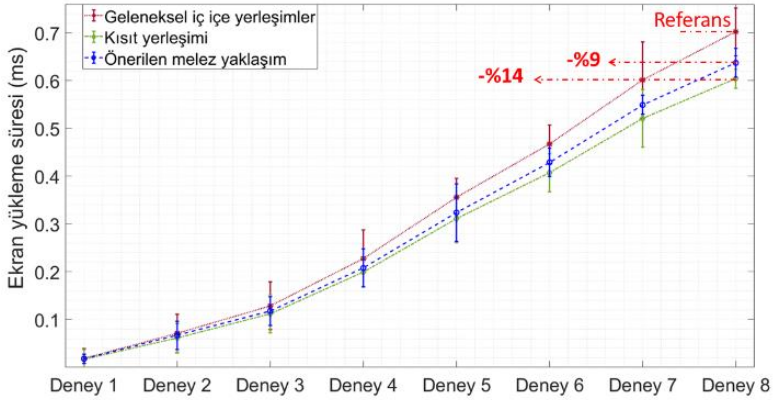
Performans değerlendirmesi sonucu elde edilen bulgular farklı cihaz donanımları için ayrı ayrı ele alınmıştır. Böylece, işletim sisteminin kullandığı donanımın işlem yeteneği gözetilerek, farklı yaklaşımlar ile tasarlanan ekranların görüntülenme süreleri ölçülmüştür. Her bir deney için ortalama ekrana yüklenme sürelerinin yanı sıra standart sapma değerleri de göz önüne alınarak, sonuçlar görselleştirilmiştir.

İlk olarak, günümüz donanım teknolojileri göz önüne alındığında, eski nesil olarak kabul gören donanım özelliklerine sahip sanal bir Android cihaz ile elde edilen bulgular Şekil 9 ile gösterilmektedir. Buna göre, Deney 1 için tasarlanan basit ve fazla bileşen içermeyen ön yüzün her üç yaklaşımda birbirine yakın sürelerde ekrana yüklendiği görülmektedir. Diğer yandan, ön yüzde bulunan eleman sayısı ve bileşen ağacının derinliği arttıkça, Deney 8'e doğru, yüklenme sürelerinin her üç yaklaşımda da arttığı gözlemlenmektedir. Bu artış, geleneksel iç içe yerleşimler kullanıldığında daha fazla olurken; Deney 8'de ön yüz yüklenme süreleri kısıt yerleşimi ile %33, önerilen melez yerleşimde ise %27 daha kısa sürmektedir. Ayrıca, her bir deney için 100 farklı çalıştırma sonucu elde edilen standart sapma değerleri, grafiklerde dikey çubuklar ile belirtilmektedir. Buna göre, farklı deneylerde elde edilen standart sapma miktarı karmaşık ön yüz tasarımlarında daha fazla olurken, deneyler boyunca %10 değerini aşmamaktadır.



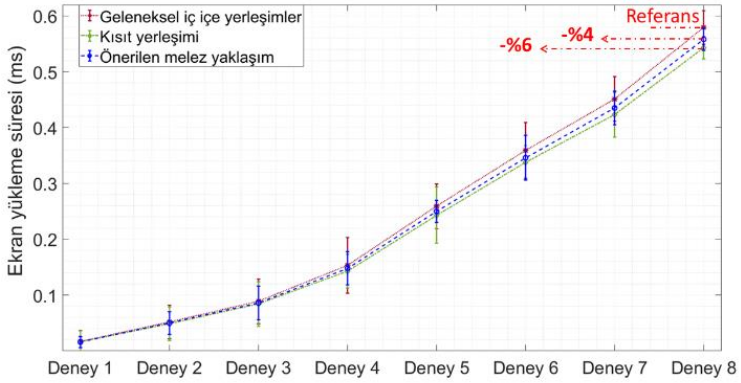
Şekil 9: Farklı deneyler için eski nesil donanım özelliklerine sahip bir cihazda elde edilen ekrana yüklenme süreleri.

İkinci olarak, günümüzde standart donanıma sahip olduğu kabul edilen bir cihaz ile tekrarlanan deneylerde, ekran yüklenme süreleri Şekil 10 ile görselleştirilmektedir. Buna göre, cihaz donanımına ilişkin işlem gücü arttıkça, ekrana yüklenme sürelerine ilişkin farklı yaklaşımlar arasındaki performans boşluğu kapanmaktadır. Dahası, eski nesil cihazda tekrarlanan deneylerden Deney 3'te bariz bir performans farkı oluşmaya başlarken, standart donanıma sahip cihazda bu fark Deney 5 için tasarlanan ön yüzde ortaya çıkmaktadır. Diğer bir deyişle, geleneksel yaklaşım ile diğer yaklaşımlar arasında bir performans farkı gözlemlemek için daha karmaşık ön yüz tasarımlarına ihtiyaç duyulmaktadır. Öte yandan, en karmaşık ön yüze sahip Deney 8 için, geleneksel yerleşimlere göre, kısıt yerleşimi ile %14; önerilen yerleşim ile %9 daha hızlı yüklenen ön yüzler geliştirmek mümkün olmuştur. Standart donanım üzerinde her bir deney için 100 kez tekrarlanan performans ölçümleri için hesaplanan standart sapma değerleri de, eski nesil donanım üzerinde tekrarlanan deneyler sırasında elde edilen standart sapma değerlerine göre azalmaktadır. Standart donanım üzerinde tekrarlanan deneylerde hesaplanan standart sapma en fazla %8 olarak Deney 7 sırasında hesaplanmıştır.



Şekil 10: Farklı deneyler için standart donanım özelliklerine sahip bir cihazda elde edilen ekrana yüklenme süreleri

Son olarak, yeni nesil donanım özelliklerine sahip bir cihazda tekrarlanan deneyler için her üç yaklaşım ile elde edilen ekrana yüklenme süresi ortalamaları ve standart sapmaları Şekil 11 ile gösterilmektedir. Buna göre, Deney 4'e kadar her bir yaklaşım birbirine son derece yakın sonuçlar üretirken; bileşen ağacı daha fazla derinleştikçe geleneksel iç içe yerleşimler ile tasarlanan ekranlar için yüklenme süresi diğer iki yaklaşım ile elde edilen ekranların yüklenme süresine göre daha yavaş kalmaktadır. Ancak bu yavaşlık, daha düşük işlem yeteneğine sahip cihazlara göre çok daha az olup, kısıt yerleşimine göre en fazla %9; önerilen yerleşime göre ise en fazla %6 seviyesindedir. Bu nedenle, Deney 8'e kadar bariz bir fark oluştuğunu söylemek mümkün değildir. Tekrarlanan deneyler için hesaplanan standart sapma değeri %5'i aşmamaktadır.



Şekil 11: Farklı deneyler için yeni nesil donanım özelliklerine sahip bir cihazda elde edilen ekrana yüklenme süreleri.

Çalışmadan elde edilen bulgulara göre, geleneksel iç içe yerleşimler kullanarak tasarlanan Android uygulama ön yüzlerinin düşük işlem yeteneğine sahip cihazlarda gecikmeli olarak yüklendiği savunulabilir. Bu yüzden, ekrana yüklenme süresinde yaşanan gecikmeler özellikle eski nesil donanımlarda daha fazla hissedilmektedir. Uygulama ekranı yüklenirken fark edilen bu durum kullanıcı deneyimini olumsuz etkilemektedir. Diğer yandan, kısıt yerleşimi kullanıldığında eski nesil donanımlarda %33'e kadar performans artışı gözlemlenirken, yeni nesil donanımlarda işlem gücü arttıkça performans farkı daha az belirginleşmektedir. Özellikle günümüzde kullanılan çok çekirdekli ve yüksek bellek donanımına sahip cihazlarda, kısıt yerleşimleri ile geleneksel iç içe yerleşimler arasındaki ön yüz yüklenme süresi farkı %6 seviyesine düşmektedir. Bu durumda, çoklu ekran desteği için yüksek donanım özelliklerine sahip bir grup cihazı destekleyen uygulamalarda, geleneksel yerleşimlerin kullanılması sonucu kullanıcı deneyiminde bir değişim gözlemlenmeyeceği savunulabilir. Diğer yandan, ön yüz tasarımı için önerilen melez yaklaşım kullanıldığında, ön yüz yüklenme süresinin geleneksel yaklaşıma göre ciddi oranda azaltıldığı savunulabilir. Bu nedenle, çoklu ekran desteği için hedeflenen donanım özellikleri göz önüne alındığında, önerilen melez yaklaşımın kısıt yerleşimi yaklaşımına tercih edilebileceği görülmektedir. Böylece, geliştiriciler için öğrenme maliyetinin, proje yaşam döngüsü için ise bakım maliyetinin azaltılması hedeflenir.

5 Sonuç

Bilgisayar donanımlarının küçülmesi ve işlem yeteneklerinin artmasıyla birlikte mobil işletim sistemleri yaygınlaşarak farklı donanım özelliklerine sahip cihazlarda kullanılmaya başlanmıştır. Android işletim sistemi farklı donanım özelliklerine sahip telefon, tablet, televizyon, saat gibi cihazlara uygulama geliştirmeye olanak sağlamak amacıyla yaygın olarak kullanılmaktadır. Diğer yandan, Android işletim sisteminin koştuğu cihaz çeşitliliği son derece fazla iken, uygulama geliştiriciler için farklı cihazlarda aynı kullanıcı deneyimini sunabilmek adına çoklu ekran desteği sorunu ortaya çıkmaktadır. Bunun çözümüne yönelik kullanılan yaklaşımlardan biri geleneksel iç içe yerleşimler ile elde edilen oransal tasarımlardır. Ancak, bu yaklaşım, ön yüz tasarımlarının ekrana yüklenmesi sırasında bileşen ağacının birçok defa taranması sebebiyle, ağaç derinliği arttıkça performans kaybına sebebiyet vermektedir. Öte yandan, 2018 yılında AndroidX sürümüyle tanıtılan kısıt yerleşimi, iç içe yerleşimlerde kullanılan fazlalık yerleşim nesneleri yerine, bileşenlerin birbirlerine göre konumlandırılmasına olanak sağlamaktadır. Kısıt yerleşimleri, ekran yüklemesi sırasında oluşan performans kaybına bir çözüm olarak görülse de, geliştiriciler için bir öğrenme maliyeti ve mevcut projelerin yaşam döngüsü için bir bakım maliyeti oluşturmaktadır. Çoklu ekran desteği ile ilgili yaklaşımların avantaj ve dezavantajları göz önüne alındığında, bu çalışmada, geleneksel iç içe yerleşimler ile kısıt yerleşimlerini birlikte kullanmayı öneren ve bileşen ağacı derinliğini belirli bir seviyede tutmayı hedefleyen melez bir ön yüz geliştirme yaklaşımı önerilmektedir.

Önerilen yaklaşım ile geliştirilen ön yüzlerin ekrana yüklenme süresi, farklı karmaşıklığa sahip 8 ön yüz tasarımı ile farklı işlem yeteneğine sahip 3 Android sanal cihazın oluşturduğu bir değerlendirme ortamında sınanarak mevcut yaklaşımlarla kıyaslanmaktadır. Elde edilen bulgulara göre, işlem yeteneği yüksek yeni nesil donanımlarda kısıt yerleşimi ile geliştirilen tasarımın, geleneksel yerleşim ile geliştirilen tasarıma göre %6 oranında daha kısa sürede ekrana yüklendiği tespit edilmektedir. Benzer şekilde, önerilen melez yaklaşım da geleneksel iç içe yerleşimlere göre %4 oranında daha hızlı ekran yükleme süreleri üretmektedir. Öte yandan, yeni nesil donanıma sahip bir cihazda tekrarlanan deneyler için ölçüm süreleri gözle algılanamayacak kadar düşük değerler içerdiğinden, gözlemlenen bu farkın kullanıcı deneyimine etki etmediği savunulabilir. İşlem yeteneği düşük eski nesil donanımlarda ise, geleneksel iç içe yerleşimler ile tasarlanan ön yüzler, kısıt yerleşimi ile tasarlanan ön yüzlere oranla %33; önerilen melez yaklaşıma göre ise %27 daha yavaş çalışmaktadır. Bu nedenle, eski nesil donanım bileşenlerine sahip cihazlarda, ön yüz yükleme süreleri daha uzun olduğundan, bu farkın kullanıcı deneyimine doğrudan etki ettiği savunulabilir. Bu etki genelde ekranın yüklenmesi sırasında donmalar şeklinde gözlemlenmektedir. Buna göre, önerilen melez yaklaşımın, eski nesil donanımlar içeren çoklu ekran desteği için uygun bir kullanıma sahip olduğu ortaya konmaktadır. Nitekim, önerilen melez yaklaşım ile elde edilen ön yüz yüklenme süreleri, kısıt yerleşimine daha yakınken, tamamen kısıt yerleşimini kullanmaya göre daha az öğrenme ve uygulama maliyetine sahiptir.

Bilgilendirme

Bu çalışmada etik kurul onay belgesine gerek yoktur.

Kaynaklar

[1] Ensmenger, N. (2018). The environmental history of computing. *Technology and culture*, 59(4), S7-S33.

[2] DeFranco, J. F. (2022). From Calculations to Computations: A Look Back at the First Computer. *Computer*, 55(1), 82-87.

[3] Teshome, A. K., Kibret, B., & Lai, D. T. (2018). A review of implant communication technology in WBAN: Progress and challenges. I*EEE reviews in biomedical engineering*, 12, 88-99.

[4] Comer, D. E. (2018). The Internet book: everything you need to know about computer networking and how the Internet works. C*hapman and Hall/CRC*.

[5] Sharma, T. N., Beniwal, M. K., & Sharma, A. (2013). Comparative study of different mobile operating systems. *International Journal of Advancements in Research & Technology, 2(3), 1-5.*

[6] Adekotujo, A., Odumabo, A., Adedokun, A., & Aiyeniko, O. (2020). A Comparative Study of Operating Systems: Case of Windows, UNIX, Linux, Mac, Android and iOS. *International Journal of Computer Applications*, 176, 16-23.

[7] Sarkar, A., Goyal, A., Hicks, D., Sarkar, D., & Hazra, S. (2019). Android application development: a brief overview of android platforms and evolution of security systems. *In 2019 Third International conference on I-SM*AC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 73-79). IEEE.

[8] Akkara, J. D., & Kuriakose, A. (2018). Innovative smartphone apps for ophthalmologists. *Kerala Journal of Ophthalmology*, 30(2), 138.

[9] Venkataraman, S., & Rajkumar, J. (2019). Development of Cell phone-Android applications for the execution of Teacher Education Curriculum. *Think India Journal*, 22(4), 9284-9289.

[10] Sweidan, S. Z., & Darabkh, K. A. (2018). VREG: A virtual reality educational game with arabic content using android smart phone. *Journal of Software Engineering and Applications*, 11(10), 500-520.

[11] Dieter, M., & Tkacz, N. (2020). The patterning of finance/security: A designerly walkthrough of challenger banking apps. *Computational Culture*, (7).

[12] Matic, A., & Gomez-Marin, A. (2019). A customizable tablet app for hand movement research outside the lab. *Journal of Neuroscience Methods*, 328, 108398.

[13] Papadakis, S., Kalogiannakis, M., & Zaranis, N. (2018). Educational apps from the Android Google Play for Greek preschoolers: A systematic review. *Computers & Education*, 116, 139-160.

[14] Maher, C. T., Oja, E., Marshall, A., Cunningham, M., Townsend, L., Worley-Hood, G., ... & Larson, A. J. (2019). Real-time monitoring with a tablet app improves implementation of treatments to enhance forest structural diversity. *Journal of Forestry*, 117(3), 280-292.

[15] Aafer, Y., You, W., Sun, Y., Shi, Y., Zhang, X., & Yin, H. (2021). Android {SmartTVs} Vulnerability Discovery via {Log-Guided} Fuzzing. *In 30th USENIX Security Symposium* (USENIX Security 21) (pp. 2759-2776).

[16] Skorupska, K., Núñez, M., Kopec, W., & Nielek, R. (2018). Older adults and crowdsourcing: Android tv app for evaluating tedx subtitle quality. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW), 1-23.

[17] Jisha, R. C., Mathews, M. P., Kini, S. P., Kumar, V., Harisankar, U. V., & Shilpa, M. (2018). An android application for school bus tracking and student monitoring system. *In 2018 IEEE International Conference on Computational Intelligence and Computing Research* (ICCIC) (pp. 1-4). IEEE.

[18] Pajic, N., & Bjelica, M. (2018). Integrating Android to Next Generation Vehicles. *In 2018 Zooming Innovation in Consumer Technologies Conference* (ZINC) (pp. 152-155). IEEE.

[19] Munasinghe, T., Patton, E. W., & Seneviratne, O. (2019). Iot application development using mit app inventor to collect and analyze sensor data. *In 2019 IEEE International Conference on Big Data (Big Data) (pp. 6157-6159).* IEEE.

[20] Özçevik, Y. (2022). Human robot interaction as a service for combatting COVID-19: an experimental case study. *Journal of Ambient Intelligence and Humanized Computing*, 1-10.

[21] Septian, I., & Alianto, R. S. (2018). Comparison analysis of android gui testing frameworks by using an experimental study. *Procedia Computer Science*, 135, 736-748.

[22] Biørn-Hansen, A., Rieger, C., Grønli, T. M., Majchrzak, T. A., & Ghinea, G. (2020). An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*, 25(4), 2997-3040.

[23] Yang, B., Xing, Z., Xia, X., Chen, C., Ye, D., & Li, S. (2021). UIS-Hunter: Detecting UI Design Smells in Android Apps. In 2021 IEEE/ACM 43rd International Conference on Software Engineering: *Companion Proceedings* (ICSE-Companion). IEEE, 89ś92.

[24] Ismail, I. (2021). New iOS 15 to offer better user experience and on-device intelligence. *New Straits Times*.

[25] Guan, J., Mao, B., & Jiang, X. (2020). The feature selection based on AndroidManifest. xml. *In Journal of Physics: Conference Series* (Vol. 1634, No. 1, p. 012027). IOP Publishing.

[26] Chen, S., Fan, L., Chen, C., Su, T., Li, W., Liu, Y., & Xu, L. (2019). Storydroid: Automated generation of storyboard for Android apps. *In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (pp. 596-607). IEEE.

[27] Mondal, S. K., Pei, Y., Dai, H. N., Kabir, H. D., & Sahoo, J. P. (2020). Boosting ui rendering in android applications. *In 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C) (pp. 285-286)*. IEEE.

[28] Hunt, J. (2021). Android Layouts and Events. In Beginner's Guide to Kotlin Programming (pp. 485-501). *Springer*, Cham.

[29] Bielik, P., Fischer, M., & Vechev, M. (2018). Robust relational layout synthesis from examples for Android. *Proceedings of the ACM on Programming Languages*, 2(OOPSLA), 1-29.

.