



**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ**

DERS: YAZILIM LAB. 1

ÖDEV: TOP EKLEME OYUNU

ÖĞRENCİLER:

SILA SERDAR, 230502014

MUHAMMED EMİR SARI, 230502054

DERS SORUMLUSU :

PROF. DR./DR. ÖĞR. ÜYESİ: Elif Pınar Hacıbeyoğlu

1 GİRİŞ

1.1 Projenin amacı

Bu projenin amacı, Python programlama dilinin standart arayüz kütüphanesi olan **Tkinter** ve resim işleme kütüphanesi olan **Pillow (PIL)** kullanarak etkileşimli bir masaüstü uygulaması geliştirmektir. Uygulama, kullanıcının belirlediği renk ve boyutlarda futbol topalarını bir tuval üzerine eklemesine ve bu topaların tuval sınırları içinde sekerek hareket etmesini sağlayan bir animasyonu kontrol etmesine olanak tanır.

Projede gerçekleştirilmesi beklenen temel işlevler şunlardır:

- Kullanıcının "football.png" adlı bir resim dosyasını temel alarak top oluşturması.
- Temel resim dosyasının, kullanıcının seçtiği renge (Kırmızı, Mavi, Sarı) göre **dinamik olarak renklendirilmesi**.
- Topların üç farklı boyutta (küçük, orta, büyük) oluşturulabilmesi.
- Oluşturulan topaların tuval üzerinde rastgele bir konumda ve rastgele bir hızla hareket etmeye başlaması.
- Topların tuval sınırlarına çarptığında gerçekçi bir şekilde sekmesi (yön değiştirmesi).
- Kullanıcının animasyonu **Başlatma**, **Durdurma** ve mevcut tüm topları **Sıfırlama** (Reset) işlevlerini kontrol edebilmesi.
- Kullanıcının mevcut tüm topaların hızını artırabilmesi ("Speed Up").
- Oluşturulan renklendirilmiş ve boyutlandırılmış resimlerin, performansı artırmak için bir **önbellek (cache)** mekanizmasında saklanması.

2 GEREKSİNİM ANALİZİ

2.1 Arayüz gereksinimleri

Kullanıcı Arayüzü Gereksinimleri:

- Uygulama, 600x500 piksel boyutlarında siyah bir ana animasyon tuvaline (Canvas) sahip olmalıdır.

- Arayüzün arka plan rengi koyu bir tema (örn. #222222) olmalıdır.
- Tuvalin altında bir kontrol paneli (Frame) bulunmalıdır.
- Kontrol panelinde, top rengini seçmek için "Kırmızı", "Mavi" ve "Sarı" seçeneklerini sunan Radyo Butonları (Radiobutton) olmalıdır.
- Kontrol panelinde, eklenecek topun boyutunu seçmek için "küçük", "orta" ve "büyük" seçeneklerini temsil eden tıklanabilir görseller (Canvas widget'ları) bulunmalıdır.
- Kontrol panelinde "START", "STOP", "RESET" ve "Speed Up" etiketli dört adet ana kontrol butonu (Button) bulunmalıdır.
- Butonların ve tıklanabilir alanların üzerine gelindiğinde (hover) görsel geri bildirim (renk değişimi) sağlanmalıdır.

Donanım Arayüzü Gereksinimleri:

- Proje standart bir masaüstü uygulaması olduğundan, klavye ve fare dışında özel bir donanım arayüzü gereksinimi bulunmamaktadır.

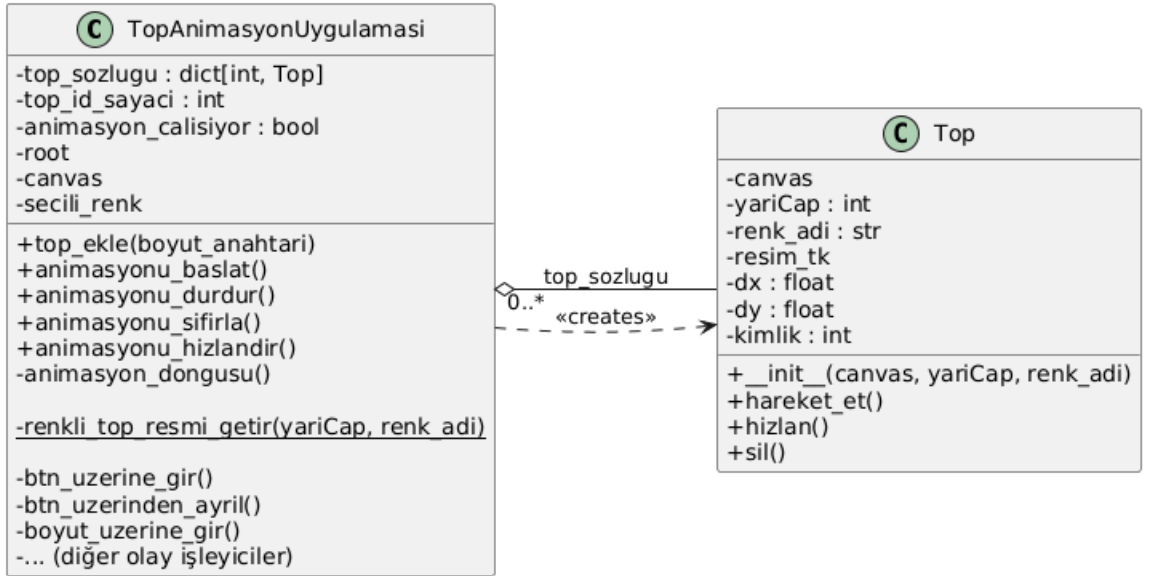
2.2 Fonksiyonel gereksinimler

- **Sistem:** Uygulama başlatıldığında football.png dosyasını ana dizinden yüklemelidir. Dosya bulunamazsa, kullanıcıyı bilgilendiren bir hata mesajı vermeli ve programı sonlandırmalıdır.
- **Resim İşleme:** Sistem, get_colored_ball_image fonksiyonu aracılığıyla, temel football.png resminin koyu renkli (siyah) piksellerini, kullanıcının Radyo Buton ile seçtiği hedef renkle değiştirmelidir.
- **Performans:** Sistem, daha önce oluşturulmuş bir (boyut, renk) kombinasyonu tekrar istendiğinde, resmi yeniden işlemek yerine IMAGE_CACHE adlı sözlükten (dictionary) çekmelidir.
- **Top Ekleme:** Kullanıcı bir boyut butonuna tıkladığında, sistem o an seçili olan renk ve seçilen boyuta göre yeni bir Ball nesnesi oluşturmalı ve balls_dict sözlüğüne eklemelidir.
- **Hareket:** "START" butonuna basıldığında, animation_loop fonksiyonu aktif hale gelmeli ve balls_dict içindeki her bir topun move() metodunu çağırmalıdır.
- **Çarpışma:** Ball nesnesinin move() metodu, topun koordinatlarının CANVAS_WIDTH veya CANVAS_HEIGHT sınırlarına ulaşp ulaşmadığını kontrol etmeli, ulaştıysa ilgili eksenindeki hızını (dx veya dy) tersine çevirmelidir (*= -1).
- **Kontrol:**
 - "STOP" butonu animation_running değişkenini False yaparak animation_loop döngüsünü durdurmalıdır.

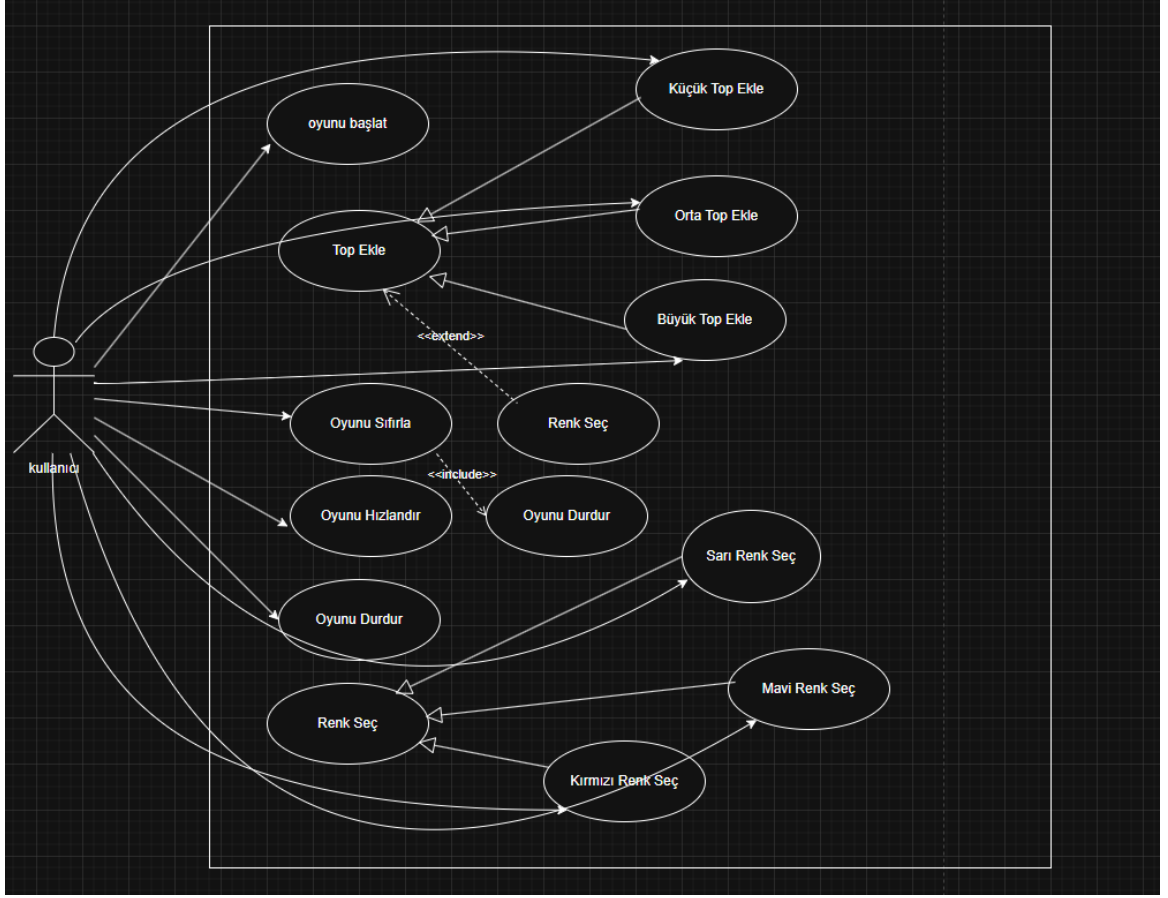
- "RESET" butonu `animation_running` değişkenini durdurmalı, `balls_dict` içindeki tüm `Ball` nesnelerinin `delete()` metodunu çağırarak onları tuvalden silmeli ve `balls_dict` sözlüğünü temizlemelidir.
- "Speed Up" butonu, `balls_dict` içindeki her topun `speed_up()` metodunu çağırarak `dx` ve `dy` değerlerini 1.25 kat artırmalıdır.

2.3 Diyagramlar

1. Class Diagram (Sınıf Diyagramı):



2. Use-Case Diagram (Kullanım Durumu Diyagramı):



3 TASARIM

3.1 Mimari tasarım

Proje, **olay güdümlü (event-driven)** bir mimari kullanılarak tasarlanmıştır.

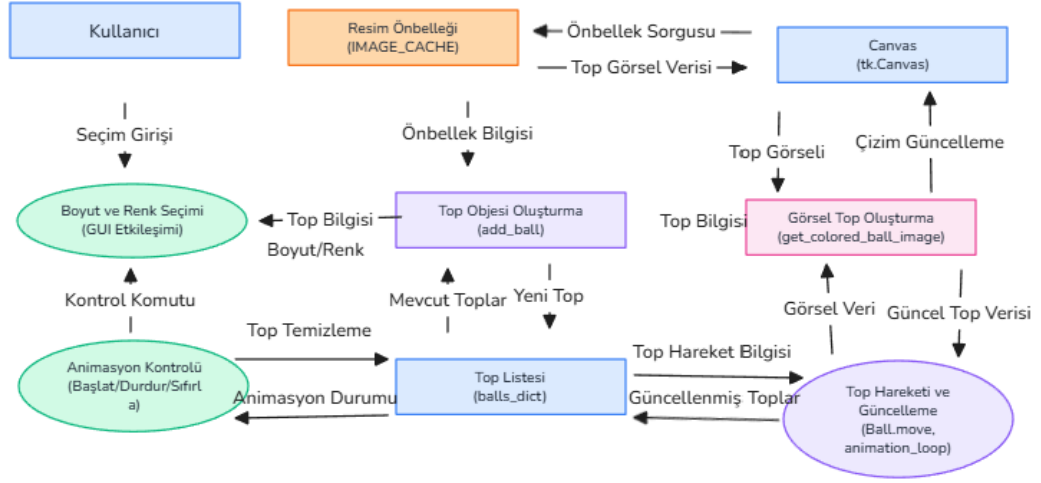
Uygulamanın ana yapısı tek bir Python dosyasında (ödev . py) yer alır ve monolitik bir yapıdadır.

- **Model-View-Controller (MVC) Benzeri Yapı:**

- **Model (Veri):** balls_dict sözlüğü, aktif olan tüm Ball nesnelerinin durumunu (konum, hız) tutar. BASE_IMAGE ve IMAGE_CACHE de veri modelinin parçalarıdır.
- **View (Görünüm):** tkinter.Tk ana penceresi, Canvas (tuval) ve kontrol Frame'leri (butonlar, radyo butonları) oluşturur. Ball sınıfının self.id'si tuval üzerindeki görsel temsildir.
- **Controller (Kontrolcü):** add_ball, start_animation, stop_animation gibi butonlara atanan fonksiyonlar, kullanıcı girdilerini (olayları) alır, Model'i (örn. balls_dict) günceller ve View'in (örn.

canvas.move) güncellenmesini tetikler. animation_loop fonksiyonu, animasyonun ana kontrolcüsü olarak görev yapar.

- **Veri Akış Diyagramı (Açıklaması):**



3.2 Kullanılacak teknolojiler

- **Programlama Dili: Python 3**
- **Kütüphaneler:**
 - **Tkinter:** Python'un standart GUI (Grafiksel Kullanıcı Arayüzü) kütüphanesidir. Pencere, tuval, butonlar, çerçeveler ve olay yönetimi (event handling) için kullanılmıştır.
 - **Pillow (PIL):** Harici bir Python kütüphanesidir (pip install Pillow). Image (resim yükleme, piksel manipülasyonu), ImageColor (renk adını RGB'ye çevirme) ve ImageTk (Pillow resimlerini Tkinter'a uyumlu hale getirme) modülleri kullanılmıştır.
 - **random:** Topların başlangıç konumlarını, yönlerini ve hızlarını rastgele belirlemek için kullanılmıştır.

3.3 Veri tabanı tasarımı

Bu projede harici bir disk tabanlı veri tabanı (SQL veya NoSQL) **kullanılmamıştır**.

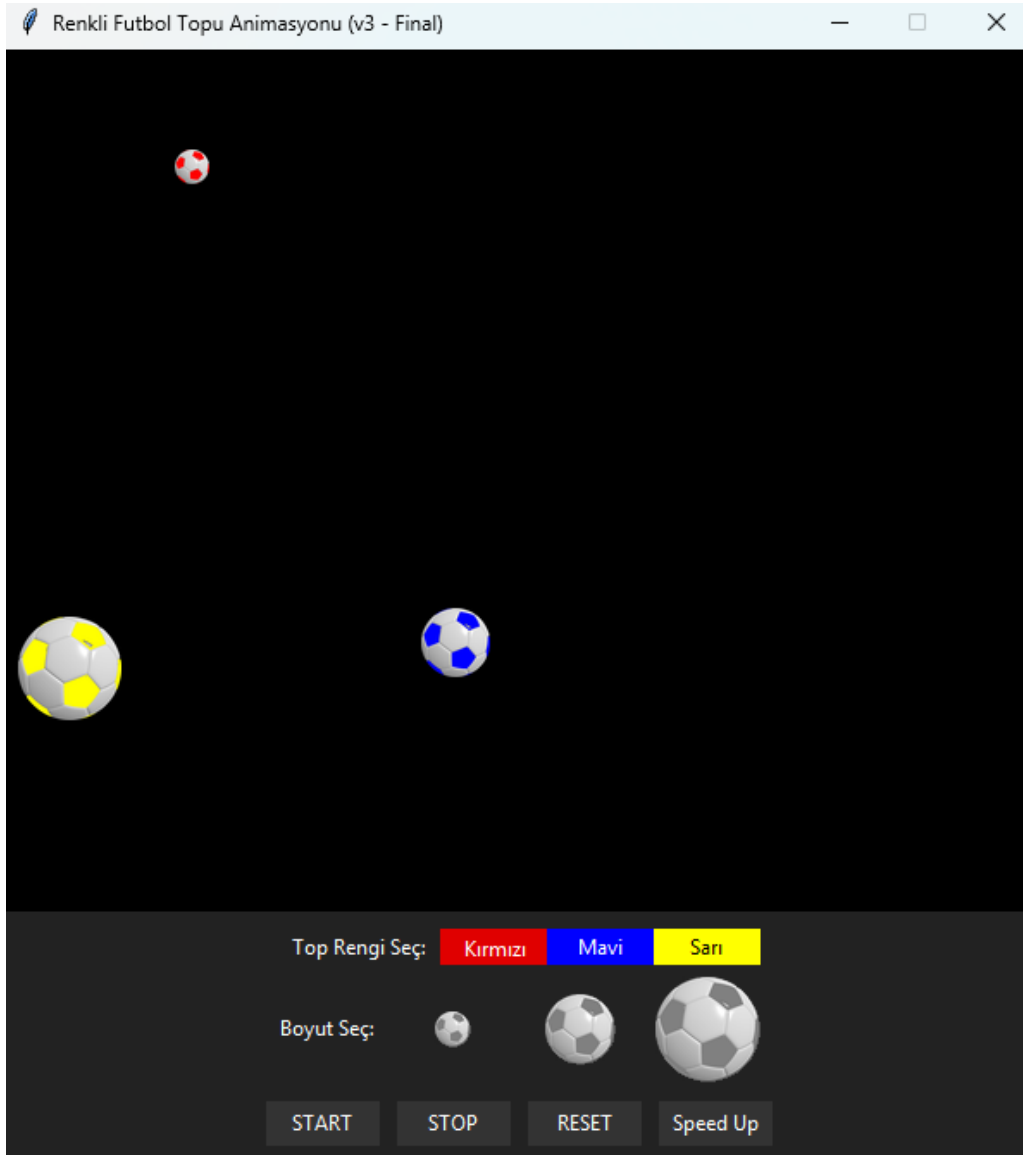
Uygulamanın çalışması süresince ihtiyaç duyulan tüm veriler (aktif topların listesi, hızları, konumları) **bellek içi (in-memory)** veri yapılarında saklanmaktadır. balls_dict adlı Python sözlüğü (dictionary), çalışan tüm Ball nesnelerini depolamak için birincil veri yapısı olarak görev yapar.

Veri tabanı kullanılmadığı için **ER Diyagramı** çizilmesine gerek duyulmamıştır.

3.4 Kullanıcı arayüzü tasarımı

Arayüz, root adı verilen ana Tkinter penceresinden oluşur. Bu pencere dikey olarak iki ana bölüme ayrılmıştır:

1. **Animasyon Alanı (Üst):** CANVAS_WIDTH (600) ve CANVAS_HEIGHT (500) boyutlarında, bg=COLOR_CANVAS (siyah) renkli bir Canvas widget'ı. Tüm top hareketleri burada gerçekleşir.



2. **Kontrol Paneli (Alt):** control_frame adlı bir Frame widget'ı. Bu panel de kendi içinde üç yatay alt bölüme ayrılır:
 - a. color_frame: Top Rengi Seçimi için üç adet Radiobutton içerir (Kırmızı, Mavi, Sarı).
 - b. add_ball_frame: Top Boyutu Seçimi için üç adet Canvas widget'ı (Küçük, Orta, Büyük top önizlemeleri) içerir.

- c. `action_button_frame`: "START", "STOP", "RESET", "Speed Up" Button widget'larını içerir.



- **Ekran Çıktısı (Açıklama):** Yukarıdaki resimde, uygulamanın çalışır hali (birden fazla renkli top hareket halindeyken) ve kontrol paneli görülmektedir.
- **Uygulamanın Çalıştırılması:**
 - Sistemde Python 3 yüklü olmalıdır.
 - Pillow kütüphanesi yüklenmelidir: `python -m pip install Pillow`
 - `ödev.py` dosyası ve `football.png` resim dosyası aynı dizinde bulunmalıdır.
 - Uygulama, terminalden `python ödev.py` komutu ile çalıştırılır.
 - Açılan penceredeki butonlar kullanılarak animasyonla etkileşime geçilir.

4 UYGULAMA

4.1 Kodlanan bileşenlerin açıklamaları

- **Ball Sınıfı:** Projenin çekirdek nesnesidir. Her bir topun yarıçapını (`radius`), rengini (`color_name`), tuval üzerindeki kimliğini (`id`), hız vektörlerini (`dx`, `dy`) ve Tkinter resim referansını (`image_tk`) yönetir. `move()` metodu, topun hareket mantığını ve duvar çarpışma tespitini içerir.
- **get_colored_ball_image() Fonksiyonu:** En karmaşık yardımcı fonksiyondur. Bir yarıçap ve renk adı alır. `BASE_IMAGE` (`football.png`) kopyalanır, pikselleri taranır; koyu renkli pikseller hedef renkle değiştirilir (`putpixel`). Ardından resim istenen çapa (`diameter`) yeniden boyutlandırılır. Performans için `IMAGE_CACHE` sözlüğünü kullanır.
- **Buton Fonksiyonları (add_ball, start_animation, stop_animation, reset_animation, speed_up_animation):** Bu fonksiyonlar, arayüzdeki butonlara bağlanan "kontrolcü" (controller) görevini üstlenirler. Global değişkenleri (`animation_running`, `balls_dict`) manipüle ederek uygulamanın durumunu yönetirler.
- **animation_loop() Fonksiyonu:** Uygulamanın "kalbidir". `animation_running` True olduğu sürece, `balls_dict` içindeki her topun `move()` metodunu çağırır

ve `root.after(16, ...)` komutuyla yaklaşık 60 FPS (saniyede 60 kare) hedefleyerek kendini tekrar çağırır.

- **Arayüz Animasyonları (`on_btn_enter`, `on_size_press vb.`):** Kullanıcı deneyimini iyileştiren küçük yardımcılardır. Fare butonların veya boyut seçicilerin üzerine geldiğinde veya tıkladığında renk değişiklikleri yaparak görsel geri bildirim sağlarlar.

4.2 Görev dağılımı

- **Bileşenlerin Tasarım ve Geliştirilmesi:**
 - Tüm bileşenlerin (Sınıf tasarımı, arayüz kodlaması, resim işleme mantığı ve animasyon döngüsü) tasarımı ve kodlaması birlikte yapılmıştır. (Sıla Serdar,230502014/Muhammed Emir Sarı,230502054)
- **Raporun Hazırlanması:**
 - Proje raporunun tüm bölümleri (Giriş, Analiz, Tasarım, Uygulama, Test) birlikte yapılmıştır. (Sıla Serdar,230502014/Muhammed Emir Sarı,230502054)

4.3 Karşılaşılan zorluklar ve çözüm yöntemleri

1. **Zorluk:** `football.png` resminin sadece siyah panellerini, seçilen renge dinamik olarak boyamak.
 - a. **Çözüm:** Pillow kütüphanesi kullanıldı. Resim piksel piksel tarandı (`getpixel`). Bir pikselin şeffaf olmadığı (`a > 0`) ve koyu bir ton olduğu (`r < 100 and g < 100 and b < 100`) kontrol edildi. Bu koşulu sağlayan pikseller, `ImageColor.getrgb(color_name)` ile alınan hedef renkle değiştirildi (`putpixel`).
2. **Zorluk:** Sürekli yeni toplar eklerken (özellikle farklı boyutlarda) resim işleme (`resize`, `recolor`) nedeniyle animasyonda takılmalar (lag) yaşanması.
 - a. **Çözüm:** Bir ön bellek (cache) mekanizması (`IMAGE_CACHE` sözlüğü) uygulandı. (`çap`, `renk_adı`) bir anahtar (key) olarak kullanıldı. Bir top resmi istenmeden önce bu sözlük kontrol edildi. Eğer resim daha önce oluşturulmuşsa, işleme adımları atlanarak doğrudan cache'den alındı; bu, performansı önemli ölçüde artırdı.
3. **Zorluk:** Tkinter'da Canvas üzerine eklenen `ImageTk.PhotoImage` nesnelerinin, fonksiyon içinden çağrıldığında çöp toplayıcı (garbage collector) tarafından silinmesi ve resimlerin görünmemesi.
 - a. **Çözüm:** Oluşturulan resim referansları (`preview_image`), kalıcı bir widget'a (örn. `size_canvas.image_reference`) veya `Ball` nesnesinin kendisine (`self.image_tk`) bir özellik olarak atanarak referanslarının korunması sağlandı.

4.4 Proje isterlerine göre eksik yönler

Projenin PDF formatında belirtilen temel isterleri (top ekleme, renklendirme, hareket, durdurma, sıfırlama, hızlandırma, duvar çarpışması) **tamamı kodlanmıştır**.

Geliştirme olarak eklenebilecek ancak mevcut isterlerde bulunmayan eksik yön şudur:

- **Topların Birbirleriyle Çarpışması:** Mevcut uygulamada toplar sadece duvarlara çarpmaktadır. Birbirlerinin içinden geçerler. Top-top çarpışma fiziği ve momentum aktarımı kodlanmamıştır.

5 TEST VE DOĞRULAMA

5.1 Yazılımın test süreci

Yazılım için ayrı bir otomatik test uygulaması (sınıf veya fonksiyon, örn. `unittest`) geliştirilmemiştir. Test süreci, geliştirici tarafından **manuel (elle) test** yöntemleriyle, kullanıcı arayüzü üzerinden gerçekleştirilmiştir.

Test edilen bileşenler ve senaryolar şunlardır:

- **Test 1 (Top Ekleme):**
 - Her renk (Kırmızı, Mavi, Sarı) seçilip her boyutta (Küçük, Orta, Büyük) top eklendi.
 - *Beklenen:* Topun doğru renk ve boyutta tuvale eklenmesi.
 - *Sonuç:* Başarılı.
- **Test 2 (Animasyon Kontrolü - Start/Stop):**
 - Tuvalde 5 top varken "START" butonuna basıldı. Toplar hareket etti.
 - "STOP" butonuna basıldı. Toplar durdu.
 - *Beklenen:* Topların hareket edip durması.
 - *Sonuç:* Başarılı.
- **Test 3 (Reset):**
 - Tuvalde 10 top varken "RESET" butonuna basıldı.
 - *Beklenen:* Tüm topların tuvalden temizlenmesi.
 - *Sonuç:* Başarılı.
- **Test 4 (Speed Up):**
 - Toplar hareket ederken 3 kez "Speed Up" butonuna basıldı.
 - *Beklenen:* Topların gözle görülür şekilde hızlanması.
 - *Sonuç:* Başarılı.
- **Test 5 (Duvar Çarpışması):**

- Tek bir top tuvalde hareket ederken dört duvara da çarpması beklendi.
- *Beklenen:* Topun her duvara çarptığında yönünü (dx veya dy) ters çevirmesi.
- *Sonuç:* Başarılı.
- **Test 6 (Hata Durumu - Dosya Yok):**
 - `football.png` dosyasının adı değiştirilerek program çalıştırıldı.
 - *Beklenen:* Programın hata mesajı vermesi ve çökmemesi (veya kontrollü sonlanması).
 - *Sonuç:* Program, terminale `HATA: 'football.png' dosyası bulunamadı.` mesajını bastı ve `exit()` ile kontrollü bir şekilde sonlandı. Başarılı.

5.2 Yazılımın doğrulanması

Manuel testler sonucunda yazılımın proje isterlerinde belirtilen **tüm temel fonksiyonları doğru ve beklediği gibi çalıştığı doğrulanmıştır.**

- **Tam ve Doğru Çalışan Bileşenler:**
 - Ball sınıfı (hareket ve çarpışma mantığı)
 - `get_colored_ball_image` (renklendirme, boyutlandırma ve önbellekleme)
 - Kontrol butonları (`start`, `stop`, `reset`, `speed_up`)
 - Top ekleme arayüzü
 - Hata yönetimi (`football.png` bulunamaması)
- **Eksik ya da Hatalı Çalışan Bileşenler:**
 - Test sürecinde kritik veya hatalı çalışan bir bileşene rastlanmamıştır.
 - **Kısıtlama (Hata Değil):** `get_colored_ball_image` fonksiyonu, `football.png` resminin *siyah panellere* sahip olduğu varsayımı üzerine kodlanmıştır. Farklı bir tasarıma sahip (örn. mavi panelli) bir top resmiyle kullanılırsa renklendirme hatalı çalışacaktır. Bu bir hata değil, mevcut resme yönelik bir tasarım tercihidir.

6. Github Bağlantısı :

<https://github.com/emirsariiii/AnimasyonluCizimEkranı>