T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ

PROJE KONUSU: ANİMASYON ÇİZİM EKRANI

ÖĞRENCİ ADI: Sude Çakır
ÖĞRENCİ NUMARASI: 230502016

ÖĞRENCİ ADI: Aybüke Sude Emek
ÖĞRENCİ NUMARASI: 230501060

DERS SORUMLUSU:

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

TARİH: 16.11.2025

1 GİRİŞ

1.1 Projenin amacı

Bu projede temel amaç; farklı boyutlarda ve renklerde topların (yuvarlakların) rastgele konumlara yerleştirilebildiği, hareket animasyonuna sahip olduğu ve kullanıcı tarafından kontrol edilebildiği **animasyonlu bir çizim ekranı** uygulamasının geliştirilmesi olmuştur. Bu amaç doğrultusunda, animasyonu yönetmek için **Start, Stop, Reset** ve **Hızlan (Speed up)** gibi butonların sisteme işlevsel olarak entegrasyonu hedeflenmiştir. **Projede gerçekleştirilmesi beklenenler:**

• Proje kapsamında bir grafik arayüz (çizim ekranı) oluşturulması gerçekleştirilmiştir. • 3 farklı boyutta top (yuvarlak) oluşturma seçeneği sisteme entegre edilmiştir. • 3 farklı renk seçeneği uygulamaya eklenmiştir.

• Topların çizim ekranına rastgele bir konuma yerleştirilmesi sağlanmıştır. • **Start** butonuna basıldığında topların rastgele bir yöne doğru harekete başlaması ve hareket animasyonuna sahip olması sağlanmıştır.

• Hareket halindeki topların çizim ekranının kenarlarına değdiğinde sekerek hareketlerine devam etmesi gerçekleştirilmiştir.

• **Stop** butonuna basıldığında topların bulundukları konumda sabit durması ve görünür kalması sağlanmıştır.

• **Reset** tuşuna basıldığında ekrandaki tüm topların silinmesi ve ekranın başlangıç konumuna getirilmesi işlevi geliştirilmiştir.

• **Hızlan (Speed up)** butonuna her basıldığında topların hareketinin bir öncekinden hızlı olacak şekilde hızlanması özelliği eklenmiştir

2 GEREKSİNİM ANALİZİ

2.1 Arayüz gereksinimleri

Proje için belirlenen kullanıcı arayüzü (UI) gereksinimleri aşağıdaki gibi analiz edilmiş ve gerçekleştirilmiştir:

• Çizim Alanı: Animasyonlu çizimlerin yapılacağı ana ekranın (Canvas) oluşturulması sağlanmıştır.

• Şekil ve Renk Seçenekleri: Kullanıcının 3 farklı boyutta top (yuvarlak) seçeneğine ve 3 farklı renk seçeneğine sahip olması sağlanmıştır. Bu seçenekler, çizim ekranına eklenecek topların niteliklerini belirlemek için kullanılmıştır. • Top Ekleme: Kullanıcının çizim ekranına farklı boyutlarda n tane top ekleyebilmesi gereksinimi karşılanmıştır.

• Kontrol Butonları: Animasyonun temel kontrolü için aşağıdaki butonların kullanıcı arayüzüne eklenmesi ve işlevsel hale getirilmesi sağlanmıştır: • Start: Topların rastgele bir yöne doğru hareket animasyonunu başlatma işlevi eklenmiştir.

• Stop: Hareket halindeki topları bulundukları konumda sabitleme ve animasyonu durdurma işlevi eklenmiştir.

• Reset: Ekranda bulunan tüm topları silerek çizim alanını başlangıç konumuna getirme işlevi eklenmiştir.

• Speed Up (Hızlan): Topların hareket hızını aşamalı olarak artırma işlevi eklenmiştir.

• Görsel Geribildirim: Animasyon süresince, topların hareket animasyonuna sahip olması ve ekran kenarlarına değdiğinde sekmeleri sağlanmıştır.

Donanım Arayüzü Gereksinimleri:

Bu proje, bir yazılım animasyon programı olduğundan, özel bir donanım arayüzü gereksinimi bulunmamaktadır.

2.2 Fonksiyonel gereksinimler

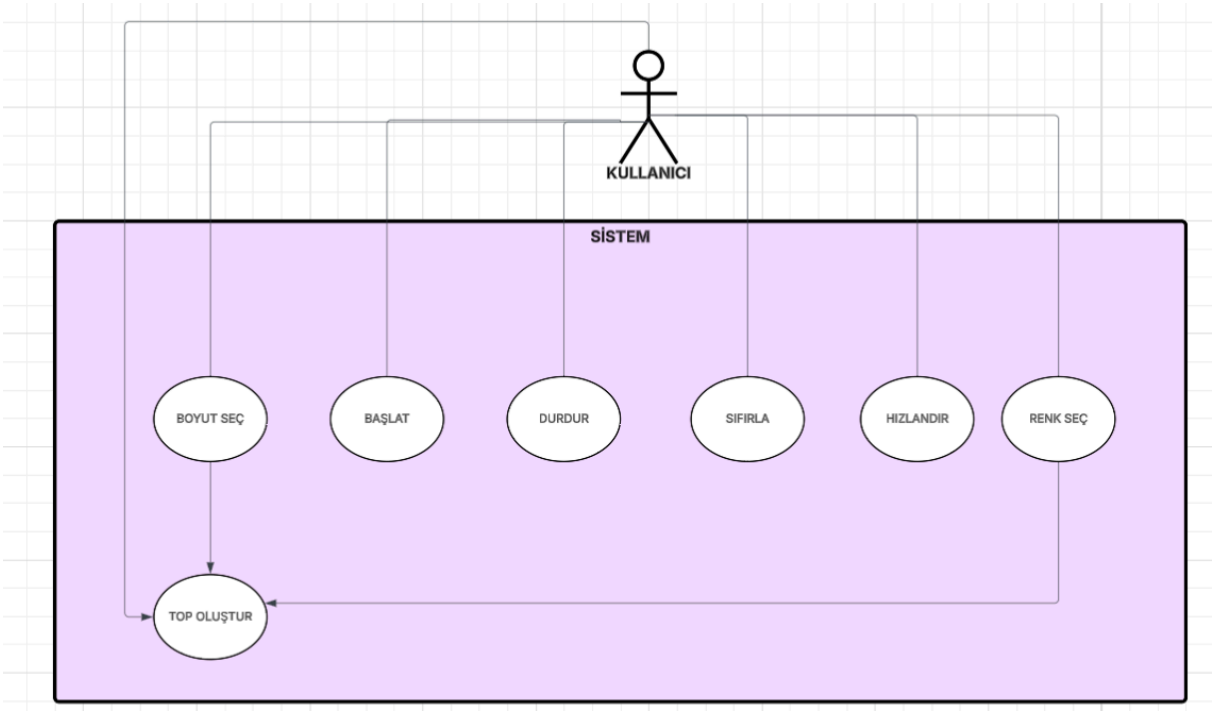
Projenin temel işlevselliğini oluşturan fonksiyonel gereksinimler, aşağıdaki maddeler halinde detaylandırılmış ve uygulama kapsamında gerçekleştirilmiştir:

- **Şekil Seçimi İşlevi:** Kullanıcının 3 farklı boyutta top (yuvarlak) seçebilmesi işlevi sağlanmıştır.
- **Renk Seçimi İşlevi:** Kullanıcının toplar için 3 farklı renk seçeneği arasından seçim yapabilmesi sağlanmıştır.
- **Top Oluşturma ve Konumlandırma:** Çizim ekranına farklı boyutlarda *n* tane top eklenebilmesi ve bu topların rastgele bir konuma yerleştirilmesi işlevi gerçekleştirilmiştir.
- **Start Fonksiyonu: Start** butonuna basıldığında, yerleştirilen topların rastgele bir yöne doğru hareket animasyonuna başlaması sağlanmıştır.
- **Kenar Çarpışma ve Sekme:** Hareket halindeki topların çizim ekranının kenarlarına değdiğinde o noktadan sekmesi ve hareketlerine devam etmesi işlevi entegre edilmiştir. • **Stop Fonksiyonu: Stop** tuşuna basıldığında, hareketin durdurulması ve topların bulundukları konumda sabitlenerek ekranda görünmeye devam etmesi sağlanmıştır. Stop durumundayken bile ekrana yeni toplar eklenebilmesi özelliği korunmuştur. • **Reset Fonksiyonu: Reset** tuşuna basıldığında, ekranda bulunan bütün topların silinmesi ve çizim ekranının başlangıç konumuna getirilmesi işlevi gerçekleştirilmiştir. • **Hızlandırma Fonksiyonu: Hızlan (Speed up)** butonuna her basıldığında, topların hareket hızının bir önceki durumdan daha hızlı olacak şekilde artırılması işlevi uygulanmıştır.

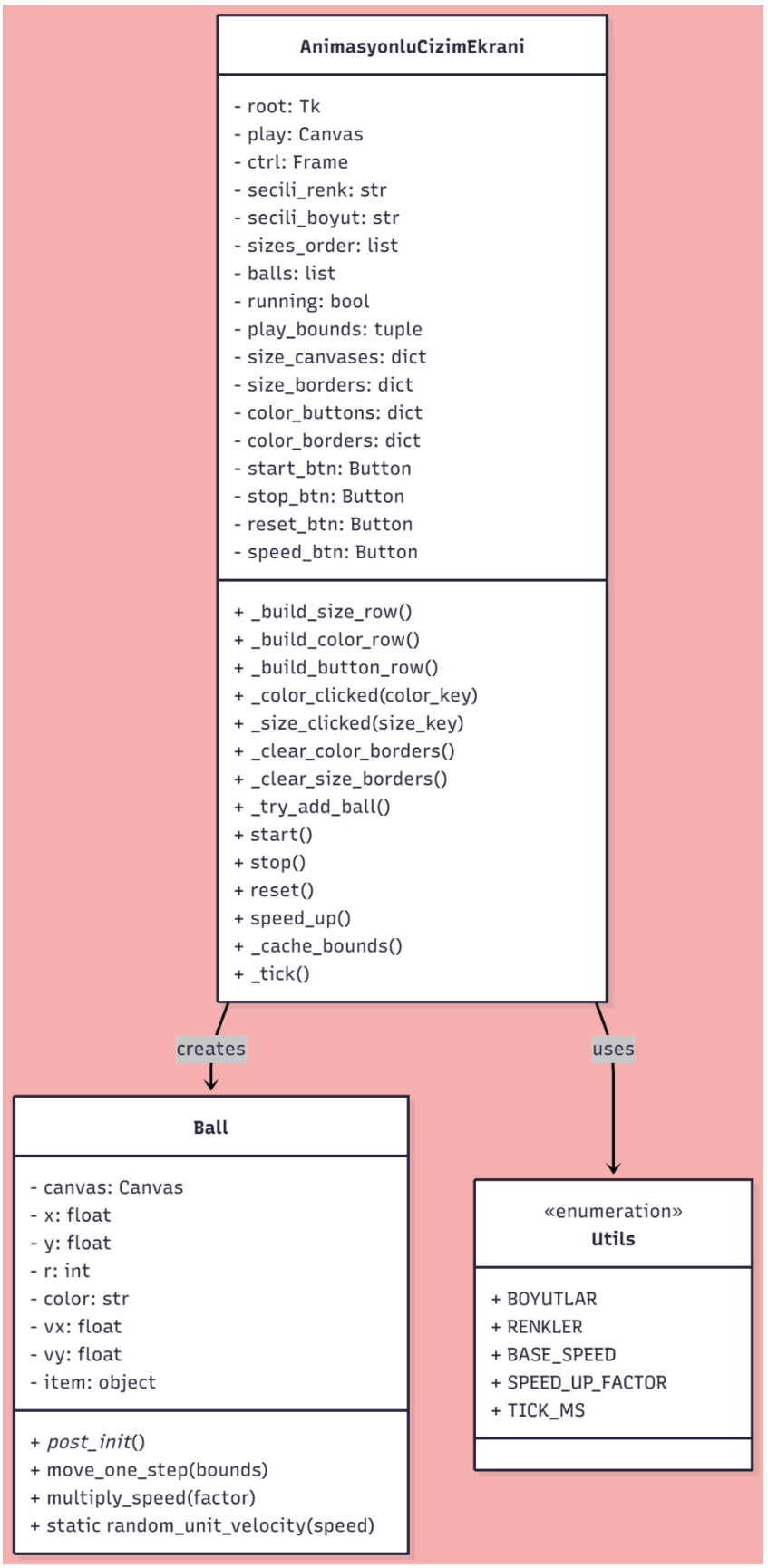
2.3 Diyagramlar

Verilen projenin gereksinimleri gereği aşağıdaki diyagramlardan uygun olan 2 tanesi rapora eklenmelidir;

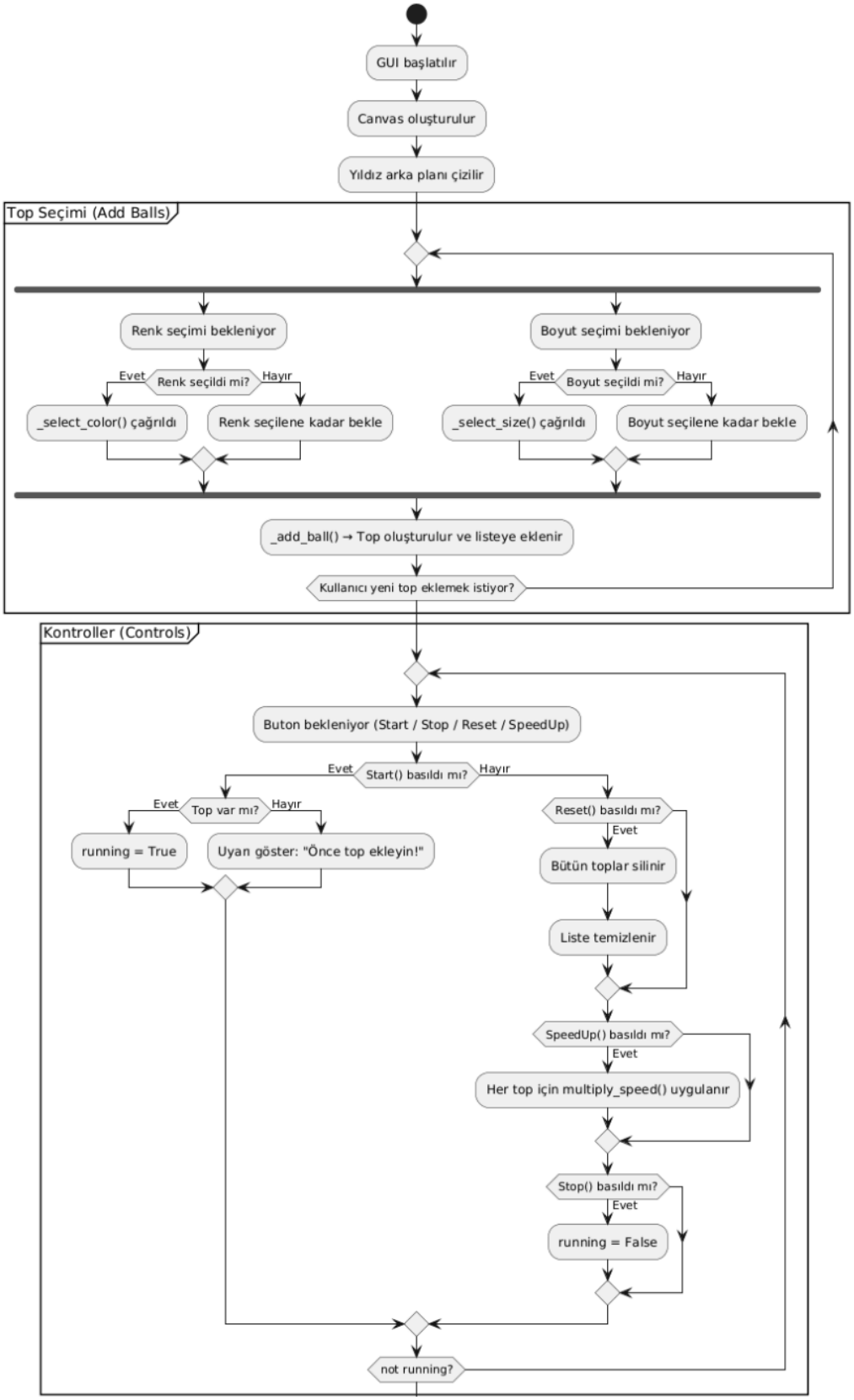
Use-Case diyagramı

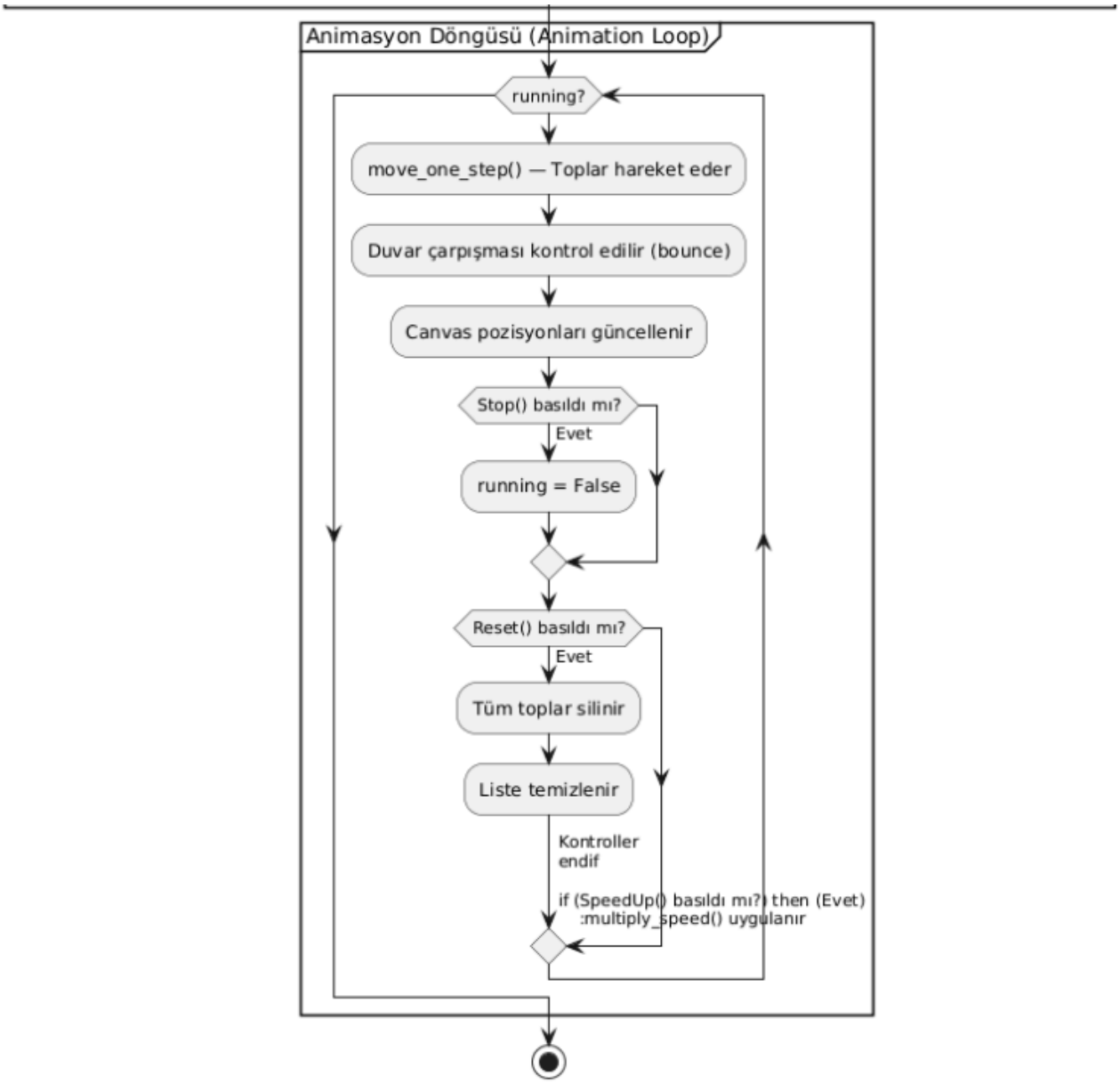


Class diyagramı



Activity diyagramı




Animasyon Döngüsü (Animation Loop)
running?
move_one_step() — Toplar hareket eder
Duvar çarpışması kontrol edilir (bounce)
Canvas pozisyonları güncellenir
Stop() basıldı mı?
Evet
running = False
Reset() basıldı mı?
Evet
Tüm toplar silinir
Liste temizlenir
Kontroller
endif
if (SpeedUp() basıldı mı?) then (Evet)
:multiply_speed() uygulanır

3 TASARIM

3.1 Mimari tasarım

Animasyonlu çizim ekranı uygulamasının mimari tasarımı, kullanıcı arayüzü, kontrol mekanizmaları ve animasyon mantığını birbirinden ayıran modüler bir yapıda tasarlanmıştır. Bu tasarım, sistemin yönetilebilirliğini ve gelecekteki olası geliştirmelere adaptasyonunu kolaylaştırmıştır.

- **Kullanıcı Arayüzü (Presentation Layer):**
  - Grafik Kullanıcı Arayüzü (GUI) bileşenleri bu katmanda konumlandırılmıştır. ○ Çizim ekranı (Canvas) ve tüm kontrol butonları (**Start, Stop, Reset, Speed Up**) bu katman kullanılarak oluşturulmuştur.
  - Kullanıcının şekil (boyut) ve renk seçimi girdilerini alması bu katmanda sağlanmıştır.
- **Kontrol ve İş Mantığı (Business/Logic Layer):**
  - Projenin temel iş mantığı ve fonksiyonel gereksinimleri bu katmanda ele alınmıştır.
  - Topların hareket yönlerinin rastgele belirlenmesi ve hız yönetimi (Speed Up) algoritmaları bu katmanda geliştirilmiştir.
  - Butonlardan gelen olayların (**Start**, **Stop**, **Reset**) işlenmesi ve animasyon döngüsünün kontrolü bu katman tarafından gerçekleştirilmiştir.
- **Veri ve Nesne Yönetimi (Data/Model Layer):**
  - Uygulamada hareket eden her bir topun (Ball) bir nesne olarak modellenmesi bu katmanda yapılmıştır.
  - Her top nesnesinin **konum**, **hız (yön)**, **boyut** ve **renk** gibi öznitelikleri bu katman aracılığıyla yönetilmiştir.
  - Topların ekran kenarlarıyla çarpışma durumunun sürekli kontrolü ve yeni bir hareket yönünün atanması bu katmanda sağlanmıştır.

- Veri Akış Diyagramı eklenmesi

3.2 Kullanılacak teknolojiler

- Yazılımın hangi dilde yazılacağı hakkında bilgi
- Kullanılacak harici kütüphaneler hakkında bilgi
- Varsa diğer teknolojiler ile ilgili açıklama

3.4 Kullanıcı arayüzü tasarımı

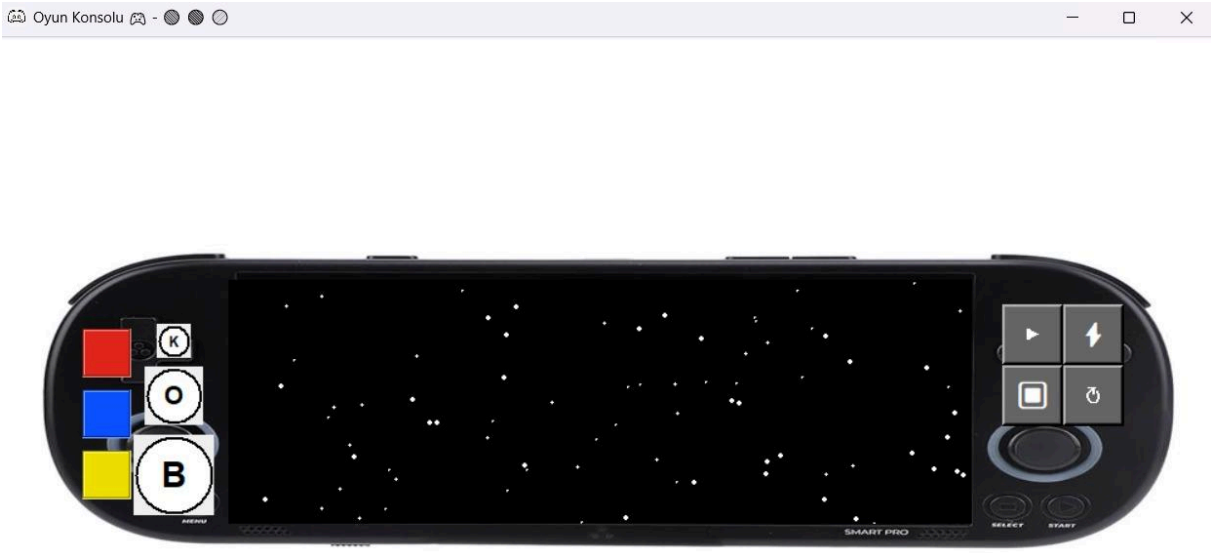
- Uygulamanın grafik arayüzü (GUI), kullanıcıların animasyonu kolayca kontrol edebileceği iki ana bölümden oluşacak şekilde tasarlanmıştır. Üst kısım **çizim ekranı** olarak ayrılmış, alt kısım ise tüm **kontrol elemanlarını** (butonlar ve şekil/renk seçenekleri) içerecek şekilde düzenlenmiştir.
- Tasarımda 3 farklı boyutta top (yuvarlak) ekleme seçeneği ve Kırmızı, Mavi, Sarı olmak üzere 3 farklı renk seçeneği görsel olarak sunulmuştur.
- Animasyon kontrolü için **START, STOP, RESET** ve **Speed Up** butonları alt panelde yer almıştır.

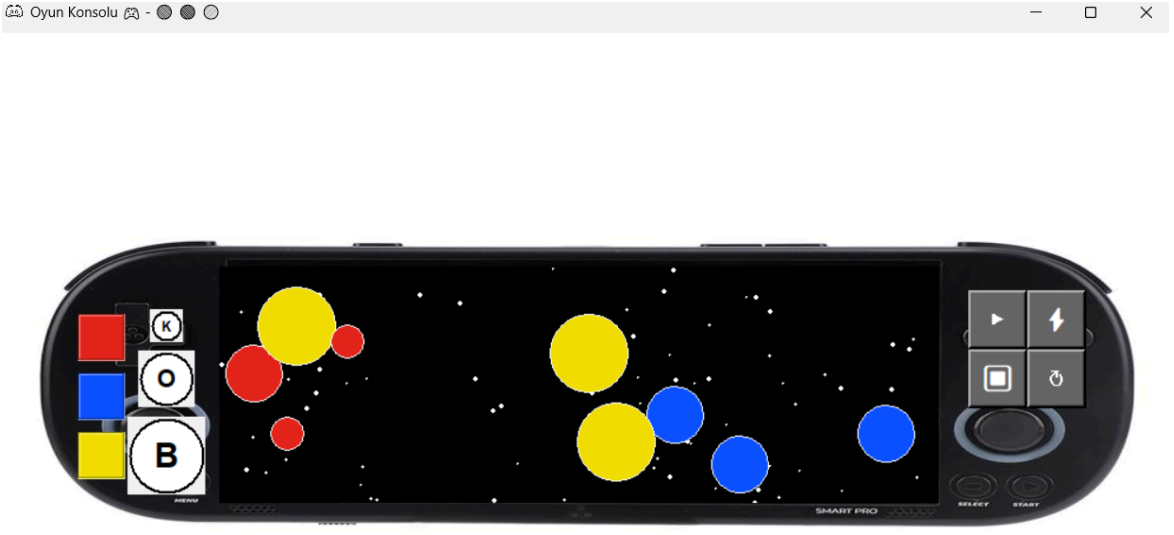
- Yazılımdan ekran çıktıları alınarak üzerinden açıklama yapılması

Uygulamanın nasıl çalıştırılacağı ile ilgili açıklama:

• Önce **top boyutu (küçük / orta / büyük)** seçilir.
• Ardından **top rengi (kırmızı / mavi / sarı)** seçilir.
• Seçimler yapıldıktan sonra **top otomatik olarak çizim ekranına eklenir.**
• **START** butonu topların rastgele yönlerde hareket etmesini sağlar.
• **STOP** butonu, topların o anki konumda durdurulmasını sağlar.
• **RESET** butonu, ekrandaki tüm topları silerek sistemi başlangıç durumuna döndürür.
• **SPEED UP** butonuna her basıldığında topların hareket hızı arttırılır.
• Kullanıcı uygulama açık olduğu sürece **istediği sayıda top eklemeye ve animasyonu kontrol etmeye devam edebilir.**

ARAYÜZ GÖRÜNTÜSÜ





topların eklenmiş hali.

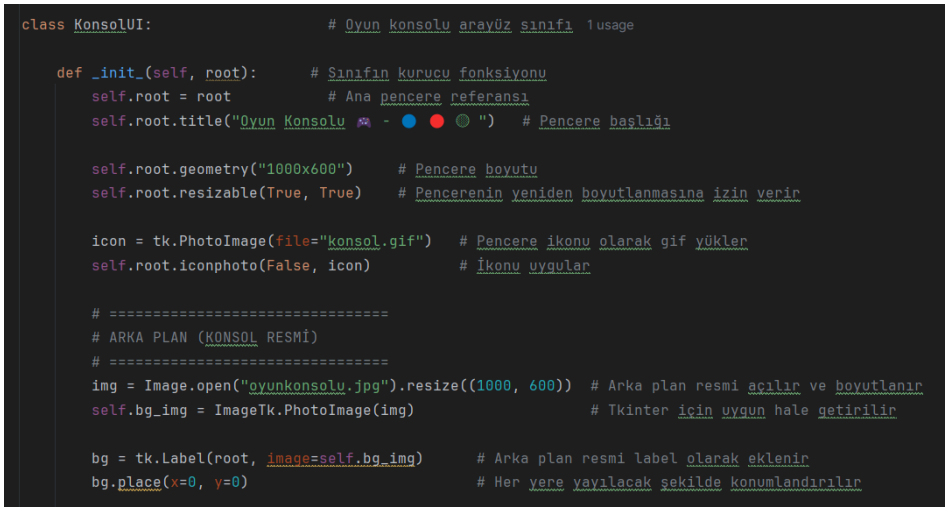
Kodlanan Bileşenlerin Açıklamaları

Bileşen	Alt Bileşen / Fonksiyon	Açıklama
Konsol UI Sınıfı	-	Oyunun grafik arayüzünü (GUI) yöneten ana sınıf. Konsol görünümünü oluşturur, butonları tanımlar ve top animasyonlarını başlatır.
Konsol UI Sınıfı	**__init__()**	Uygulama başlatıldığında çalışan kurucu fonksiyon. Görevleri: pencere boyutunu ve ikonunu ayarlar, arka plan görselini çizer, oyun alanı (Canvas) oluşturur, rastgele yıldız arka plan çizer, sağ tarafta kontrol butonlarını (Start, Stop, Reset, SpeedUp) oluşturur, sol tarafta renk ve boyut seçim butonlarını oluşturur, seçilen renk/boyutları saklamak için değişkenleri başlatır, tüm topları depolamak için liste oluşturur ve animasyon döngüsünü _tick() ile başlatır.
Konsol UI Sınıfı	**toggle_fullscreen()**	F11'e basıldığında ekranı tam ekran moduna alır veya çıkarır.
Konsol UI Sınıfı	**exit_fullscreen()**	Escape tuşuna basıldığında tam ekran modundan çıkarır.
Konsol UI Sınıfı	**play_click()**	Tıklama sesini çalar.
Konsol UI Sınıfı	**_select_color(color_key)**	Kullanıcı bir renk seçtiğinde çalışır; seçili rengi kaydeder ve _add_ball() ile top

		eklemeyi dener.
Konsol UI Sınıfı	**_select_size(size_key)**	Kullanıcı bir boyut seçtiğinde çalışır; seçili boyutu kaydeder ve _add_ball() ile top ekler.
Konsol UI Sınıfı	**_add_ball()**	Top oluşturabilmek için renk ve boyut seçilmiş olmalıdır. Seçili renk/boyut yoksa çıkar. Topun başlangıç konumunu rastgele belirler, rastgele yönlü hız üretir, Ball sınıfı ile top nesnesi oluşturur, toplar listesine ekler ve seçimleri sıfırlar.
Konsol UI Sınıfı	**start()**	Start butonuna basıldığında çalışır. Hiç top yoksa uyarı verir; varsa animasyonu başlatır (running=True).
Konsol UI Sınıfı	**stop()**	Stop butonuna basıldığında çalışır. Top yoksa veya Start'a basılmadan Stop'a basılırsa kullanıcıyı bilgilendirir. Animasyonu durdurur (running=False).
Konsol UI Sınıfı	**reset()**	Reset butonuna basıldığında çalışır. Top yoksa bilgi mesajı gösterir; varsa tüm topları ekrandan siler ve Ball listesini temizler.
Konsol UI Sınıfı	**speed_up()**	Speed Up butonuna basıldığında tüm topların hızını artırır. Animasyon çalışmıyorsa uyarı verir. Her top için multiply_speed() çağrılır.

Konsol UI Sınıfı	**_tick()**	Oyun animasyonunun kalbidir. Yaklaşık her 16 ms'de (60 FPS) çalışır. running=True ise her top için move_one_step() çağrılır ve döngü sürekli tekrar eder.
Ball Modülü	-	Topların hareketi, çarpışması ve hız güncellenmesi bu sınıf tarafından yönetilir.
	random_unit_velocity(speed)	Rastgele bir açı üretir ve verilen hızda (BASE_SPEED) x ve y hız vektörlerini döndürür.
	__post_init__()	Ball nesnesi oluşturulduktan sonra otomatik çalışır. Canvas üzerinde topun oval görselini yaratır.
	move_one_step(bounds)	Topu verilen sınırlar içinde bir adım hareket ettirir. x ve y konumunu hızlara göre günceller, sınırları aşarsa duvardan seker ve canvas üzerinde güncel konumu yeniden çizer.
	multiply_speed(factor)	Topun hızını belirli bir katsayıyla çarpar. Speed Up butonunda çalışır.
utils.py Modülü	-	Oyunda kullanılan sabit değerleri içerir.
utils.py Modülü	**TICK_MS**	Animasyon yenileme hızı (ms). 16 ms ≈ 60 FPS.

utils.py Modülü	**BASE_SPEED**	Topların başlangıç hızını belirler.
utils.py Modülü	**SPEED_UP_FACTOR**	Speed Up butonuna basıldığında hızın kaç kat artacağını belirler.
utils.py Modülü	**RENKLER**	Kullanılabilir renkleri (kırmızı, mavi, sarı) HEX kodlarıyla tutar.
utils.py Modülü	**BOYUTLAR**	Top boyutlarının yarıçap değerlerini içerir (küçük-orta-büyük).



```python
class KonsolUI:                  # Oyun konsolu arayüz sınıfı  1 usage

    def _init_(self, root):      # Sınıfın kurucu fonksiyonu
        self.root = root         # Ana pencere referansı
        self.root.title("Oyun Konsolu 🎮 - 🔵 🔴 ⚫ ")   # Pencere başlığı

        self.root.geometry("1000x600")     # Pencere boyutu
        self.root.resizable(True, True)    # Pencerenin yeniden boyutlanmasına izin verir

        icon = tk.PhotoImage(file="konsol.gif")   # Pencere ikonu olarak gif yükler
        self.root.iconphoto(False, icon)          # İkonu uygular

        # ================================
        # ARKA PLAN (KONSOL RESMİ)
        # ================================
        img = Image.open("oyunkonsolu.jpg").resize((1000, 600))  # Arka plan resmi açılır ve boyutlanır
        self.bg_img = ImageTk.PhotoImage(img)                    # Tkinter için uygun hale getirilir

        bg = tk.Label(root, image=self.bg_img)      # Arka plan resmi label olarak eklenir
        bg.place(x=0, y=0)                          # Her yere yayılacak şekilde konumlandırılır
```

Bu Python kod parçası, **Tkinter** kütüphanesini kullanarak bir **"Oyun Konsolu"** arayüz sınıfı olan KonsolUI'ın başlangıç ayarlarını yapar.Sınıf, ana pencerenin boyutunu **1000x600** olarak belirler, başlık, ikon ve yeniden boyutlandırma izinlerini ayarlar; ayrıca **'oyunkonsolu.jpg'** dosyasından arka plan resmini yükleyip pencereye yerleştirir.

```python
    # ================================
    # ARKA PLAN (KONSOL RESMİ)
    # ================================
    img = Image.open("oyunkonsolu.jpg").resize((1000, 600))  # Arka plan resmi açılır ve boyutlanır
    self.bg_img = ImageTk.PhotoImage(img)                    # Tkinter için uygun hale getirilir

    bg = tk.Label(root, image=self.bg_img)      # Arka plan resmi label olarak eklenir
    bg.place(x=0, y=0)                          # Her yere yayılacak şekilde konumlandırılır

    # ================================
    # TOPLARIN HAREKET ALANI (EKRAN)
    # ================================
    self.screen_x = 190      # Oyun ekranının x konumu
    self.screen_y = 200      # Oyun ekranının y konumu
    self.screen_w = 610      # Ekranın genişliği
    self.screen_h = 200      # Ekranın yüksekliği

    self.play = tk.Canvas(       # Topların hareket edeceği canvas
        root,
        bg="black",              # Arkaplan siyah (oyun ekranı)
        highlightthickness=0     # Kenarlık kapalı
    )
    self.play.place(             # Canvası belirtilen koordinatlara yerleştirir
        x=self.screen_x,
        y=self.screen_y,
        width=self.screen_w,
        height=self.screen_h
```

Bu kod parçası, arayüzün (**KonsolUI**) **arka plan resmi** ayarlandıktan sonra, **topların hareket edeceği** alanı tanımlar.

```python
# ================================
# SAĞ TARAFTAKİ TUŞLAR
# (Start, Speed, Stop, Reset)
# ================================
def yuvarlak_buton(x, y, text, command):   # Ortak buton oluşturucu fonksiyon
    btn = tk.Button(
        root,
        text=text,           # Butonun üzerinde yazacak karakter
        command=command,     # Buton basılınca çağrılacak fonksiyon
        bg="#666666",        # Gri arka plan
        fg="white",          # Beyaz yazı
        borderwidth=3,       # Kenarlık kalınlığı
        relief="raised",     # Kabartmalı görünüm
        font=("Arial", 16, "bold"),
        width=4,             # Boyut
        height=2
    )
    btn.place(x=x - 25, y=y - 25, width=50, height=50)  # Tam yuvarlak görünüm yaratır

# Sağ tuş grubunun yerleşimi
yuvarlak_buton(x=850, y=245, text="▶", command=self.start)      # Start
yuvarlak_buton(x=900, y=245, text="⚡", command=self.speed_up)   # Speed Up
yuvarlak_buton(x=850, y=295, text="■", command=self.stop)       # Stop
yuvarlak_buton(x=900, y=295, text="↻", command=self.reset)      # Reset
```

Bu kod parçası, arayüzün sağ tarafında yer alacak olan **Start, Speed Up, Stop** ve **Reset** butonlarını oluşturur.

```python
# ================================
# SOL TARAFTAKİ RENK & BOYUT TUŞLARI
# ================================
def renk_buton(x, y, color_key):          # Renk butonu oluşturucu
    btn = tk.Button(
        root,
        bg=RENKLER[color_key],            # Renk
        command=lambda k=color_key: self._select_color(k),
        borderwidth=2,
        relief="groove"
    )
    btn.place(x=x - 20, y=y - 20, width=40, height=40)   # Kare buton konumu

def boyut_buton(x, y, size_key):      # Boyut butonu (içinde daire olan canvas)
    radius = BOYUTLAR[size_key]       # Boyutun yarıçapı
    diameter = radius * 2             # Çap

    btn = tk.Canvas(
        root,
        width=diameter,
        height=diameter,
        bg="#f0f0f0",
        highlightthickness=0
    )

    circle = btn.create_oval(          # Daire çizimi
        2, 2, diameter - 2, diameter - 2,
        fill="white",
        outline="black",
```

Bu kod parçası, kullanıcı arayüzünün **sol tarafındaki** renk ve boyut seçimi için gerekli butonları tanımlar.

```python
    def _init_(self, root):      # Sınıfın kurucu fonksiyonu
        def boyut_buton(x, y, size_key):      # Boyut butonu (içinde daire olan canvas)
            def on_click(event=None):         # Tıklama animasyonu
                btn.itemconfig(circle, fill="#cccccc")   # Daireyi gri yap
                self._select_size(size_key)              # Boyutu seç
                btn.after( ms: 120, lambda: btn.itemconfig(circle, fill="white"))  # Eski haline dön

            btn.bind("<Button-1>", on_click)   # Tıklama olayı bağlanır

            btn.place(x=x - radius, y=y - radius)   # Ekrana yerleştirme

        # Boyut düğmeleri
        boyut_buton( x: 145,  y: 250,  size_key: "kucuk")
        boyut_buton( x: 145,  y: 295,  size_key: "orta")
        boyut_buton( x: 145,  y: 360,  size_key: "buyuk")

        # Renk düğmeleri
        renk_buton( x: 90,  y: 260,  color_key: "kirmizi")
        renk_buton( x: 90,  y: 310,  color_key: "mavi")
        renk_buton( x: 90,  y: 360,  color_key: "sari")

        # ================================
        # ANİMASYON DEĞİŞKENLERİ
        # ================================
```

Bu kod parçası, daha önce tanımlanan **boyut ve renk butonu** fonksiyonlarını kullanarak arayüze buton ekler.

```python
def toggle_fullscreen(self, event=None):  1 usage
    self.fullscreen = not self.fullscreen           # Tersine çevir
    self.root.attributes("-fullscreen", self.fullscreen)

def exit_fullscreen(self, event=None):  1 usage
    self.fullscreen = False
    self.root.attributes("-fullscreen", False)

def play_click(self):  4 usages
    winsound.PlaySound( sound: "tiklamasesi.wav", winsound.SND_FILENAME | winsound.SND_ASYNC)

# ================================
# RENK & BOYUT SEÇİMİ
# ================================
def _select_color(self, color_key):  1 usage
    self.play_click()          # Ses oynat
    self.secili_renk = color_key  # Rengi kaydet
    self._add_ball()              # Eğer boyut da seçildiyse top ekle

def _select_size(self, size_key):  1 usage
    self.play_click()          # Ses
    self.secili_boyut = size_key  # Boyutu kaydet
    self._add_ball()              # Eğer renk de seçiliyse top ekle

def _add_ball(self):  2 usages
    if self.secili_renk is None or self.secili_boyut is None:
        return                 # Renk + Boyut ikisi de seçilmeden top eklenmez
```

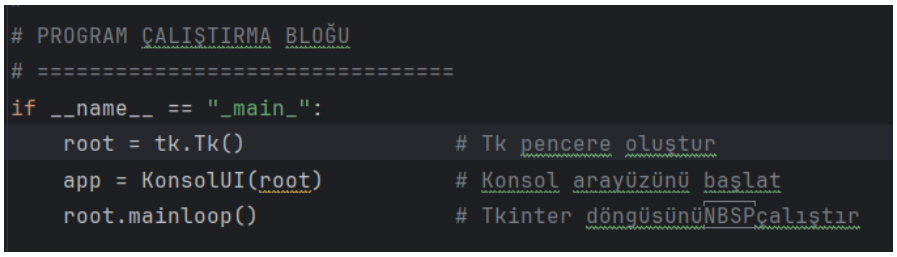
Bu kod parçası, **ekran yönetimi, ses çalma** ve **renk/boyut seçimini** yöneten yardımcı metotları içerir.**hem renk hem de boyut seçilmişse** ekrana yeni bir top eklemeyi dener.

```python
def _add_ball(self):  2 usages
    if self.secili_renk is None or self.secili_boyut is None:
        return                  # Renk + Boyut ikisi de seçilmeden top eklenmez

    r = BOYUTLAR[self.secili_boyut]      # Çap/yarıçap belirlenir
    xmin, ymin = 0, 0                    # Sınırlar
    xmax, ymax = self.screen_w, self.screen_h

    x = random.uniform(r, xmax - r)      # Rastgele bir x konumu
    y = random.uniform(r, ymax - r)      # Rastgele bir y konumu

    vx, vy = Ball.random_unit_velocity(BASE_SPEED)  # Rastgele hız yönü
    b = Ball(self.play, x, y, r, RENKLER[self.secili_renk], vx, vy)   # Top oluştur

    self.balls.append(b)         # Listeye ekle
    self.secili_renk = None      # Seçimleri sıfırla
    self.secili_boyut = None

# ================================
# TUŞLAR (START, STOP, RESET, SPEED UP)
# ================================
def start(self):  1 usage
    self.play_click()
    if len(self.balls) == 0:             # Hiç top yoksa çalışmaz
        messagebox.showwarning( title: "Uyarı",  message: "Lütfen önce top ekleyiniz!")
        return
    self.running = True                  # Animasyon başlasın

def stop(self):  1 usage
    self.play_click()
```

Bu kod parçası, **top ekleme** ve **oyun animasyonu kontrol** metotlarını içerir.

```python
def reset(self):  1 usage
    if self.running == False and len(self.balls) == 0:   # Zaten boşsa
        messagebox.showinfo( title: "Uyarı",  message: "Ekranda sıfırlanacak top yok!")
        return

    self.running = False                 # Animasyonu durdur
    for b in self.balls:                 # Tüm topları sil
        self.play.delete(b.item)
    self.balls.clear()                   # Listeyi temizle

def speed_up(self):  1 usage
    winsound.PlaySound( sound: "hizsesi.wav", winsound.SND_FILENAME | winsound.SND_ASYNC)

    if not self.running:     # Toplar duruyorsa hızlanamaz
        messagebox.showwarning( title: "Uyarı",  message: "Lütfen önce START butonuna basınız!")
        return

    for b in self.balls:     # Tüm topların hızını arttır
        b.multiply_speed(SPEED_UP_FACTOR)

# ================================
# ANİMASYON TİCK FONKSİYONU
# ================================
def _tick(self):  2 usages
    if self.running:                                 # Eğer animasyon aktifse
        for b in self.balls:
            b.move_one_step((0, 0, self.screen_w, self.screen_h))  # Her topu bir adım hareket ettir

    self.root.after(TICK_MS, self._tick)            # Kendini tekrar çağırır (sonsuz loop)
```

Bu kod parçası, **oyunun sıfırlanması, top hızlarının artırılması** ve **ana animasyon döngüsünün** yönetilmesini sağlar.



```
# PROGRAM ÇALIŞTIRMA BLOĞU
# ================================
if __name__ == "_main_":
    root = tk.Tk()                  # Tk pencere oluştur
    app = KonsolUI(root)            # Konsol arayüzünü başlat
    root.mainloop()                 # Tkinter döngüsünüNBSPçalıştır
```

Bu kod parçası, programın ana başlangıç bloğudur.pencerenin ekranda görünmesini ve kullanıcı etkileşimlerine (buton tıklamaları, fare hareketleri vb.) yanıt vermesini sağlayan Tkinter olay döngüsünü başlatır.

4.2Görev dağılımı

Proje, gereksinim analizi, mimari tasarım ve tüm uygulama bileşenlerinin geliştirilmesi aşamaları dahil olmak üzere, iki kişilik ekip tarafından **tamamen ortaklaşa** yürütülmüştür. Tasarım ve kodlama süreçlerinin her ikisi de, ekip üyeleri arasında dengeli ve sürekli işbirliği ile gerçekleştirilmiştir. Fonksiyonel gereksinimlerin (Start-Stop, hızlandırma, çarpışma algoritmaları) kodlanması ve projenin raporlanması süreci de ortak sorumlulukla yürütülmüş, bu sayede projenin tüm aşamalarında tutarlılık sağlanmıştır.

4.3**Karşılaşılan zorluklar ve çözüm yöntemleri**

- Geliştirme sürecinde karşılaşılan problemler ve bunlara getirilen çözümler

4.4**Proje isterlerine göre eksik yönler**

- Proje gerçeklenmesi beklenen görevlerden hiç kodlanamayanlar varsa belirtilmesi

5 TEST VE DOĞRULAMA

5.1**Yazılımın test süreci**

Proje tamamlandıktan sonra uygulamanın doğru çalışıp çalışmadığını görmek amacıyla çeşitli test senaryoları uygulanmıştır. Bu süreçte öncelikle top boyutu ve renk seçimi adımlarının doğru işlediği gözlemlenmiş, kullanıcının yaptığı seçimlerin çizim ekranına doğru şekilde yansıtıldığı doğrulanmıştır. Ardından animasyon kontrol butonları test edilmiş; **START** komutunun topları harekete geçirdiği, **STOP** komutunun ise topları mevcut konumlarında durdurduğu kontrol edilmiştir. **RESET** butonuna basıldığında ekrandaki tüm topların silindiği ve uygulamanın başlangıç durumuna döndüğü doğrulanmıştır. **SPEED UP** işlevinin de topların hareket hızını kademeli olarak artırdığı test edilmiştir. Ek olarak, toplar hareket halindeyken yeni bir top eklendiğinde bu topun da animasyona dahil olduğu ve hareket etmeye başladığı gözlemlenmiştir. Pencere boyutu değiştirilerek yapılan testlerde ise arayüzün ölçeklendirmeyi doğru şekilde uyguladığı ve görüntü bütünlüğünün korunduğu görülmüştür. Tüm testler farklı senaryolarla tekrar edilerek uygulamanın tutarlı çalıştığı doğrulanmıştır.

5.2 Yazılımın doğrulanması

Gerçekleştirilen testlerin sonucunda yazılımın proje gereksinimlerini tam olarak yerine getirdiği doğrulanmıştır. Uygulama, kullanıcı etkileşimlerine sorunsuz şekilde yanıt vermekte, toplar rastgele yönlerde hareket ederken ekran sınırlarına çarptıklarında doğru sekme davranışını göstermektedir. Start, Stop, Reset ve Speed Up fonksiyonlarının birbirleriyle uyumlu ve kararlı biçimde çalıştığı görülmüştür. Test sürecinde tespit edilen iki önemli durum, ekip çalışması ile giderilmiştir: hareket halindeyken eklenen topların başlangıçta hareketsiz kalma sorunu çözülmüş ve arayüz yeniden boyutlandırıldığında oluşan orantı bozulması giderilmiştir. Bu düzenlemeler sonrasında uygulama hem işlevsel açıdan hem de kullanıcı deneyimi bakımından iyileştirilmiştir. Sonuç olarak yazılımın **kararlı, amacına uygun ve kullanılabilir** olduğu doğrulanmıştır.

6 GitHub Bağlantıları

Projenin GitHub linki öğretim elemanlarının erişebileceği şekilde rapor içinde paylaşılmalıdır.

https://github.com/aybukesude/Ball-animation-game

https://github.com/sude4141/Oyun_Konsolu