



**T.C**  
**KOCAELİ SAęLIK VE TEKNOLOJİ ÜNİVERSİTESİ**  
**MÜHENDİSLİK VE DOęA BİLİMLERİ FAKÜLTESİ**  
**BİLGİSAYAR MÜHENDİSLİęİ**

**PROJE KONUSU: ANİMASYONLU ÇİZİM EKRANI**

**ÖęRENCİ ADI: DİLA SERAY TEGÜN**  
**ÖęRENCİ NUMARASI: 220501022**  
**GİTHUB: <https://github.com/dilaseray>**

**DERS SORUMLUSU:**  
**PROF. DR./DR. ÖęR. ÜYESİ ELİF PINAR HACİBEYOęLU**

**TARİH: 24.11.2024**

# 1. GİRİŞ

## 1.1 Projenin Amacı

- Bu projenin amacı, kullanıcıya basit bir animasyon uygulaması geliştirme yeteneğini kazandırmaktır. Kullanıcılar, farklı boyutlarda ve renklerde topları ekranda hareket ettirerek GUI (Grafiksel Kullanıcı Arayüzü) geliştirme becerilerini geliştirirler.
- Projede, kullanıcıya top ekleme, hareketi başlatma, durdurma, hızlandırma ve ekranı sıfırlama gibi işlemleri yapabileceği bir arayüz sunulması beklenmektedir.
- Projede gerçekleştirilmesi beklenenler:
  - Kullanıcının belirlediği boyut ve renkte topları ekleyebilme.
  - Topların rastgele yönde hareket etmesini sağlayan bir animasyon.
  - Topların duvarlara çarpıp sekmesini sağlama.
  - Stop, Reset ve Speed Up gibi kontrol butonlarıyla kullanıcı kontrolü sağlamak.

## 2. GEREKSİNİM ANALİZİ

### 2.1 Arayüz Gereksinimleri

- Kullanıcı arayüzü, basit ve kullanışlı olacak şekilde tasarlanmıştır. Top ekleme, hareket başlatma, durdurma, sıfırlama ve hızlandırma gibi işlemler için butonlar yer alır.
- Donanım arayüzü gereksinimi olarak, bu uygulamanın bir fare ve klavye ile kontrol edilmesi yeterlidir.

### 2.2 Fonksiyonel Gereksinimler

- Kullanıcı, ekranda farklı boyutlarda ve renklerde top oluşturabilmelidir.
- Kullanıcı, topların hareketini başlatabilmeli ve durdurabilmelidir.
- Kullanıcı, ekrandaki topları sıfırlayabilmelidir.
- Kullanıcı, topların hızını artırabilmelidir.

## 3. TASARIM

### 3.1 Mimari Tasarım

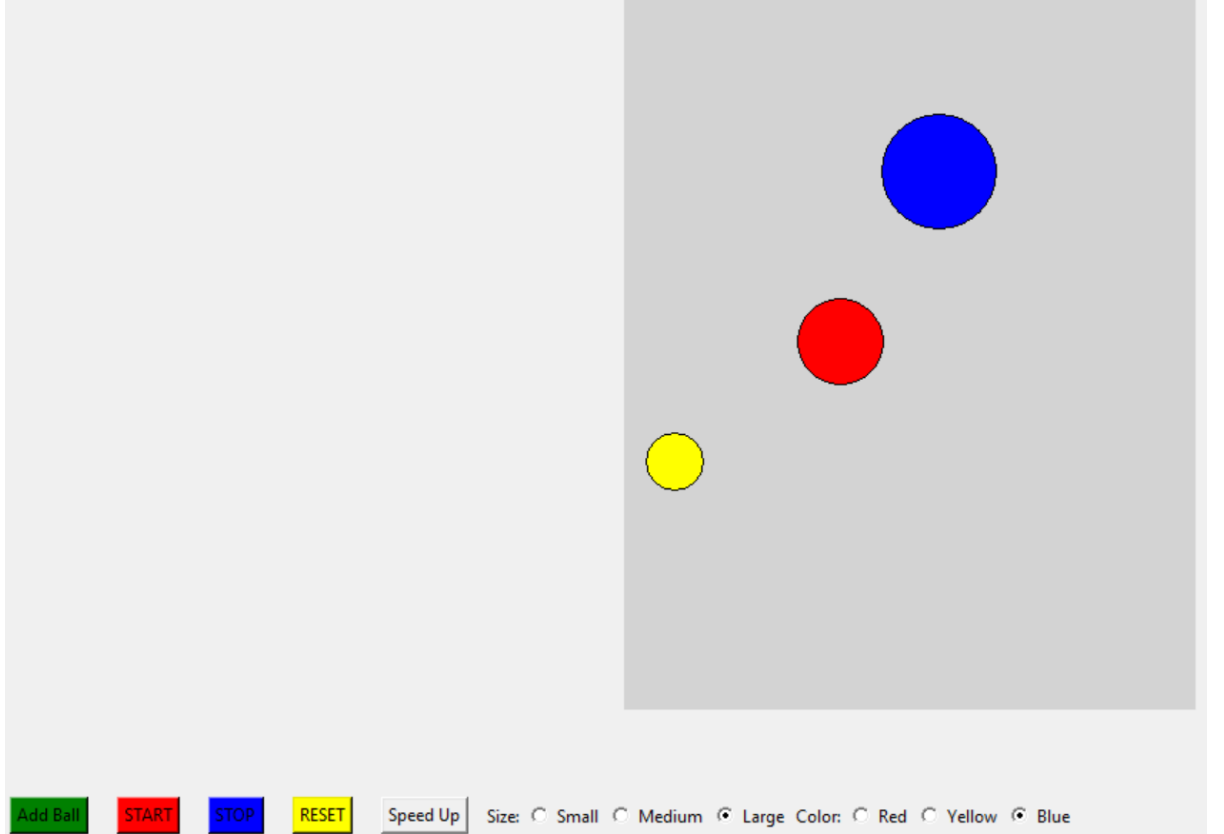
- Uygulamanın ana bileşenleri:
  - Kullanıcı Arayüzü: Tkinter kullanılarak oluşturulmuş bir grafiksel arayüz.
  - Top Nesneleri: Canvas üzerinde oluşturulan ve hareket ettirilen toplar.
  - Kontrol Mekanizması: Butonlarla kullanıcının animasyonu kontrol etmesi.

### 3.2 Kullanılacak Teknolojiler

- Yazılım Python dilinde geliştirilmiştir.
- Tkinter kütüphanesi, grafiksel arayüzü oluşturmak için kullanılmıştır.
- Rastgele konum ve yön için "random" kütüphanesi kullanılmıştır.

### 3.3 Kullanıcı Arayüzü Tasarımı

- Kullanıcı arayüzü, ekranda topların görüldüğü bir canvas ve kontrol butonlarından oluşmaktadır.



## 4. UYGULAMA

### 4.1 Kodlanan Bileşenlerin Açıklamaları

```
1 import tkinter as tk
2 import random
```

- tkinter: Python'un standart GUI kütüphanesi. Kullanıcı arayüzü oluşturmak için kullanılıyor.
- random: Topların rastgele konumlara yerleştirilmesi ve rastgele yönlere hareket etmesi için kullanılıyor.

```

4      class BallAnimation:
5          def __init__(self, root):
6              self.root = root
7              self.root.title("Ball Animation")

```

- BallAnimation: Topların animasyonunu ve kullanıcı arayüzünü yöneten sınıf.
- self.root.title("Ball Animation"): Pencerenin başlığını "Ball Animation" olarak ayarlıyor.

```

9          # Canvas creation
10         self.canvas = tk.Canvas(root, width=400, height=500, bg="lightgray")
11         self.canvas.pack()

```

- self.canvas: Topların çizileceği alanı oluşturur (canvas). Bu alanın genişliği 400, yüksekliği ise 500 pikseldir ve arka plan rengi açık gridir (lightgray).

```

14         self.add_ball_button = tk.Button(root, text="Add Ball", command=self.add_ball, bg="green")
15         self.add_ball_button.pack(side=tk.LEFT, padx=10)
16
17         self.start_button = tk.Button(root, text="START", command=self.start_animation, bg="red")
18         self.start_button.pack(side=tk.LEFT, padx=10)
19
20         self.stop_button = tk.Button(root, text="STOP", command=self.stop_animation, bg="blue")
21         self.stop_button.pack(side=tk.LEFT, padx=10)
22
23         self.reset_button = tk.Button(root, text="RESET", command=self.reset_canvas, bg="yellow")
24         self.reset_button.pack(side=tk.LEFT, padx=10)
25
26         self.speed_up_button = tk.Button(root, text="Speed Up", command=self.speed_up)
27         self.speed_up_button.pack(side=tk.LEFT, padx=10)

```

- Bu butonlar, kullanıcı arayüzünde topları eklemek (Add Ball), animasyonu başlatmak (START), durdurmak (STOP), sıfırlamak (RESET) ve hızlandırmak (Speed Up) için kullanılır.
- command parametresi, butona tıklandığında hangi fonksiyonun çağrılacağını belirtir.

```

30         self.balls = []
31         self.running = False
32         self.speed = 5

```

- self.balls: Tüm topları ve onların hareket bilgilerini saklayan liste.
- self.running: Animasyonun çalışıp çalışmadığını izlemek için kullanılır.
- self.speed: Topların başlangıç hızı.

```

39 tk.Label(root, text="Size:").pack(side=tk.LEFT)
40 tk.Radiobutton(root, text="Small", variable=self.size_var, value=20).pack(side=tk.LEFT)
41 tk.Radiobutton(root, text="Medium", variable=self.size_var, value=30).pack(side=tk.LEFT)
42 tk.Radiobutton(root, text="Large", variable=self.size_var, value=40).pack(side=tk.LEFT)
43
44 # Color selection
45 tk.Label(root, text="Color:").pack(side=tk.LEFT)
46 tk.Radiobutton(root, text="Red", variable=self.color_var, value="red").pack(side=tk.LEFT)
47 tk.Radiobutton(root, text="Yellow", variable=self.color_var, value="yellow").pack(side=tk.LEFT)
48 tk.Radiobutton(root, text="Blue", variable=self.color_var, value="blue").pack(side=tk.LEFT)

```

- Kullanıcı, boyut (Small, Medium, Large) ve renk (Red, Yellow, Blue) seçeneklerinden birini seçerek top ekleyebilir.
- Radiobutton widget'ları, kullanıcıya belirli seçenekler sunar.

```

50 def add_ball(self):
51     # Create a ball with user-selected size and color
52     size = self.size_var.get()
53     color = self.color_var.get()
54     self.create_ball(size, color)
55
56     1 usage
57     def create_ball(self, size=None, color=None):
58         # Create a ball with specified size and color
59         if size is None:
60             size = random.choice([20, 30, 40])
61         if color is None:
62             color = random.choice(["red", "green", "blue"])
63         x = random.randint(size, 400 - size)
64         y = random.randint(size, 500 - size)
65         ball = self.canvas.create_oval(x - size, y - size, x + size, y + size, fill=color)
66         dx = random.choice([-self.speed, self.speed])
67         dy = random.choice([-self.speed, self.speed])
68         self.balls.append((ball, dx, dy))

```

- add\_ball(): Kullanıcının seçtiği boyut ve renk ile yeni bir top oluşturur.
- create\_ball(size, color): Belirtilen boyut ve renkte rastgele bir konumda bir top oluşturur. Topun hareket yönü ve hızı rastgele ayarlanır.

```

69     def start_animation(self):
70         if not self.running:
71             self.running = True
72             self.animate()
73
74     1 usage
75     def stop_animation(self):
76         self.running = False
77
78     1 usage
79     def reset_canvas(self):
80         self.running = False
81         for ball, _, _ in self.balls:
82             self.canvas.delete(ball)
83         self.balls = []
84
85     1 usage
86     def speed_up(self):
87         self.speed += 2
88         # Update the speed of all existing balls
89         self.balls = [(ball, random.choice([-self.speed, self.speed]),

```


- start\_animation(): Animasyonu başlatır. self.running True yapılarak animasyonun başlaması sağlanır.
- stop\_animation(): Animasyonu durdurur.
- reset\_canvas(): Tüm topları ve animasyonu sıfırlar.
- speed\_up(): Topların hareket hızını artırır. Bu hız her tıklamada bir öncekinin üzerine eklenir.

```

88     def animate(self):
89         if self.running:
90             for i in range(len(self.balls)):
91                 ball, dx, dy = self.balls[i]
92                 coords = self.canvas.coords(ball)
93                 if coords[0] <= 0 or coords[2] >= 400:
94                     dx = -dx
95                 if coords[1] <= 0 or coords[3] >= 500:
96                     dy = -dy
97                 self.canvas.move(*args: ball, dx, dy)
98                 self.balls[i] = (ball, dx, dy)
99
100         self.root.after(50, self.animate)

```

- animate(): self.running True olduğu sürece topların hareketini sağlar.
- Topların kenarlara çarpması durumunda hareket yönlerini değiştirir (sekme efekti).
- Her 50 milisaniyede bir self.animate() fonksiyonunu tekrar çağırarak animasyonu devam ettirir.

```
102  if __name__ == "__main__":  
103     root = tk.Tk()  
104     app = BallAnimation(root)  
105     root.mainloop()
```

- `root = tk.Tk()`: Ana pencere oluşturulur.
- `app = BallAnimation(root)`: `BallAnimation` sınıfından bir uygulama örneği oluşturulur.
- `root.mainloop()`: Pencerenin sürekli olarak ekranda kalmasını ve kullanıcı etkileşimlerine cevap vermesini sağlar.

### 4.3 Karşılaşılan Zorluklar ve Çözüm Yöntemleri

- Topların canvas sınırlarında sekmesini sağlamak bazen beklenmeyen davranışlar gösterdi. Bu sorun, topun koordinatlarını kontrol eden mantığı düzenleyerek çözüldü.
- Hareketli topları durdurma ve sıfırlama işleminde senkronizasyon problemi yaşandı. Bu sorun, `self.running` değişkenini kullanarak kontrol edildi.

### 4.4 Proje İsterlerine Göre Eksik Yönler

- Projede eksik kalan herhangi bir özellik bulunmamaktadır. Tüm istenen özellikler başarıyla gerçekleştirilmiştir.

## 5. TEST VE DOĞRULAMA

- Aşağıdaki görselde test sürecinde ekranda topların hareketi gözlemlenmektedir:

