**T.C**

**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ** **LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

**YAZILIM MÜHENDİSLİĞİ PROGRAMI**

**ÖDEV KONUSU**

**ANİMASYONLU ÇİZİM EKRANI**

**Hazırlayan**

**Talha Tuna Emrah Şahin**

**220502015 220502025**

**DERS SORUMLUSU**

**Dr. Öğr Üyesi Elif Pınar HACIBEYOĞLU**

**İÇİNDEKİLER**

**24/11/2024**

[1. ÖZET 3](#_Toc183342645)

[2. GİRİŞ 3](#_Toc183342646)

[3. YÖNTEM 3](#_Toc183342647)

[3.1 Class Ball: 4](#_Toc183342648)

[3.2 Class BallAnimation: 5](#_Toc183342649)

[4. SONUÇ VE ÖĞRENİLEN DERSLER 9](#_Toc183342650)

[5. GİTHUB LİNKLERİ 9](#_Toc183342651)

# ÖZET

Bu proje, kullanıcıların farklı boyutlarda ve renklerde toplar ekleyerek animasyon deneyimi yaşamalarını sağlamaktadır. Rastgele yerleştirilen toplar, ekran üzerinde hareket eder ve sınırlarla etkileşime geçtiklerinde sekme hareketi sergiler. Hareketler, Start, Stop ve Reset butonlarıyla kontrol edilebilirken, hızlanma özelliği sağlayan Speed Up butonu animasyon dinamiklerini artırır.

# GİRİŞ

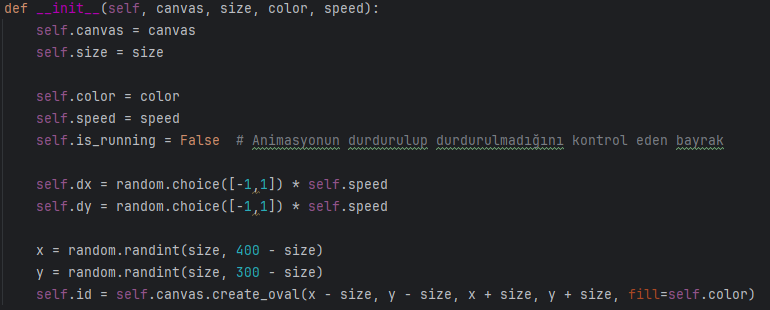
Bu çalışma, Python programlama dili kullanılarak bir dinamik bir top animasyonu geliştirmeyi hedeflemektedir. Kullanıcıların etkileşimli bir şekilde top ekleyebildiği, hızlandırabildiği ve animasyonu başlatıp durdurabildiği bir sistem oluşturulmuştur. Kod, temel olarak Python'un standart kütüphanelerinden biri olan tkinter kullanılarak tasarlanmış ve basit animasyon teknikleriyle görselleştirilmiştir.

# YÖNTEM

Bu yazılımı Python programlama dili ve “tkinter” kütüphanesini kullanarak geliştirdik. Animasyonu gerçekleştirmek için iki ana sınıf tanımladık: “Ball” ve “BallAnimation”. Ball sınıfı, topların hareketini ve tuval üzerindeki davranışlarını kontrol etmekte, BallAnimation sınıfı ise kullanıcı arayüzünü, animasyon kontrol mekanizmalarını ve topların genel yönetimini sağlamaktadır. Hareket animasyonunu, tkinter’ın after metodunu kullanarak periyodik olarak güncelledik; topların hız ve yön bilgisi ise rastgele atanarak hareketlilik sağladık.

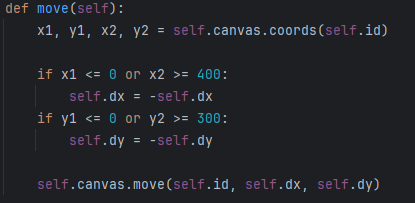
## Class Ball:

**def\_\_init\_\_:**



Sınıfın \_\_init\_\_ metodu, topun özelliklerini (kanvas, boyut, renk ve hız) ayarlar. is\_running özelliği, animasyonun durdurulup durdurulmadığını kontrol eder ve başlangıçta False olarak ayarlanır. Topun hareket yönü, dx ve dy ile belirlenir ve bu değerler rastgele bir şekilde seçilip hıza göre ayarlanır. Top, kanvas üzerinde rastgele bir başlangıç konumuna yerleştirilir ve belirtilen renk, boyut ve konumda bir oval olarak çizilir. Bu yapı, grafiksel bir ortamda hareket edebilen bir top animasyonu oluşturmak için kullanılabilir.

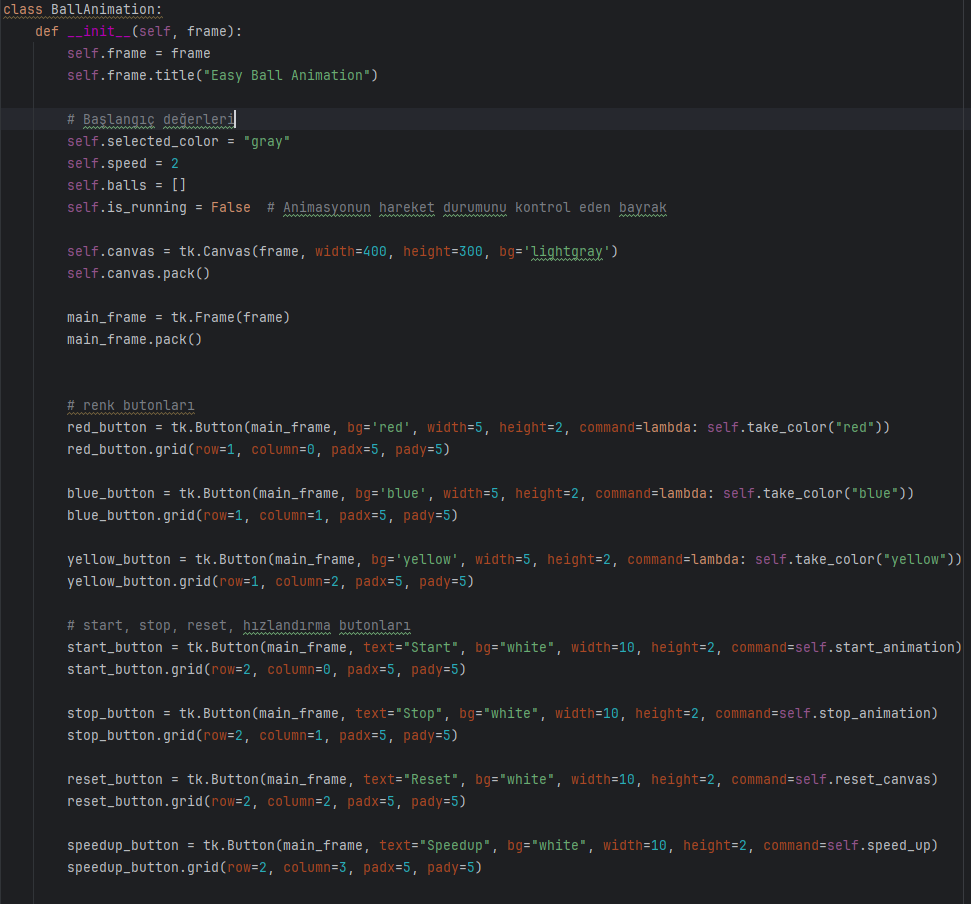
**def move(self):**

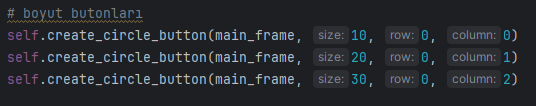


Bu fonksiyon, bir topun konumunu günceller ve hareketini sağlar. İlk olarak, self.canvas.coords(self.id) ile topun mevcut koordinatlarını alır. Eğer topun yatay kenarları (x1 veya x2) tuvalin sınırlarına çarparsa (x1 ≤ 0 veya x2 ≥ 400), self.dx ters çevrilir, yani yatay hareket yönü değiştirilir. Eğer topun dikey kenarları (y1 veya y2) tuvalin sınırlarına çarparsa (y1 ≤ 0 veya y2 ≥ 300), self.dy ters çevrilir, yani dikey hareket yönü değiştirilir. Son olarak, self.canvas.move(self.id, self.dx, self.dy) çağrılarak top, mevcut hız ve yön değerlerine göre hareket ettiril

## Class BallAnimation:

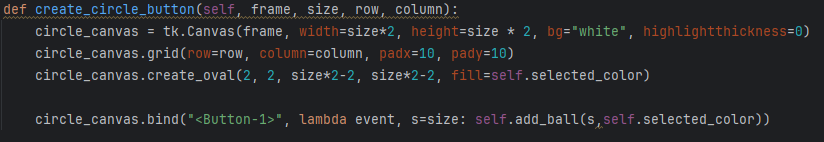
**def \_\_init\_\_(self,frame):**





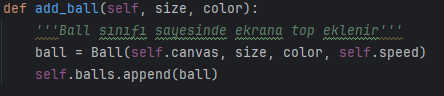
Bu kod parçası, BallAnimation sınıfının \_\_init\_\_ (yapıcı) fonksiyonudur ve uygulamanın başlangıçta yapılan ayarlarını yapar. Fonksiyon şu işlemleri gerçekleştirir: İlk olarak, self.selected\_color = "gray" ile seçilen top rengini gri olarak ayarlar, self.speed = 2 ile top hızını 2 olarak belirler, self.balls = [] ile boş bir liste oluşturur, burada topların nesneleri tutulacaktır ve self.is\_running = False ile animasyonun başta durduğunu belirten bir bayrak ayarlar. Sonra, self.canvas = tk.Canvas(frame, width=400, height=300, bg='lightgray') ile 400x300 boyutlarında ve açık gri arka planlı bir tuval (Canvas) oluşturur ve self.canvas.pack() ile bu tuvali pencereye ekler. main\_frame = tk.Frame(frame) ile butonlar ve diğer bileşenler için bir ana çerçeve oluşturur, main\_frame.pack() ile de bu ana çerçeveyi pencereye yerleştirir. Ardından, üç renk butonu (kırmızı, mavi, sarı) oluşturur, her biri self.take\_color() fonksiyonunu çağırarak renk seçim işlemini gerçekleştirir. Son olarak, animasyonu kontrol etmek için Start, Stop, Reset ve Speedup butonları oluşturur ve her biri sırasıyla start\_animation(), stop\_animation(), reset\_canvas(), speed\_up() fonksiyonlarını tetikler. Boyut seçmek için 10, 20 ve 30 piksel boyutunda üç buton ekler ve her biri create\_circle\_button() fonksiyonunu çağırarak top ekleme işlemini tetikler.

**def create\_circle\_button(self, frame, size, row, column):**



Bu fonksiyon, top eklemek için boyut seçimi yapan butonları oluşturur. İlk olarak, circle\_canvas = tk.Canvas(frame, width=size\*2, height=size\*2, bg="white", highlightthickness=0) ile, belirli bir boyutta (size) beyaz bir Canvas (tuval) oluşturur. Bu tuvalde, boyutlara göre yuvarlak bir şekil çizilecektir. highlightthickness=0 parametresi, Canvas'ın kenarlıklarını kaldırır. Sonra, circle\_canvas.grid(row=row, column=column, padx=10, pady=10) ile bu Canvas'ı belirli bir satır ve sütunda yerleştirir. Ayrıca, tuvalin etrafında 10 piksellik boşluk bırakılır. Ardından, circle\_canvas.create\_oval(2, 2, size\*2-2, size\*2-2, fill=self.selected\_color) ile, size boyutunda bir oval (yuvarlak) şekil çizilir. Bu oval, Canvas'ın içinde beyaz zemin üzerine, self.selected\_color renginde olur. Son olarak, circle\_canvas.bind("<Button-1>", lambda event, s=size: self.add\_ball(s,self.selected\_color)) ile, Canvas'a tıklanması olayını bağlar. Kullanıcı tuvale tıkladığında, self.add\_ball(s, self.selected\_color) fonksiyonu çağrılır ve size (boyut) ile seçilen renk (self.selected\_color) parametre olarak gönderilir. Bu işlem, ekranda top ekleme işlevini başlatır.

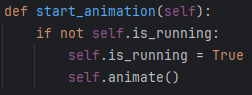
**def add\_ball(self, size, color):**

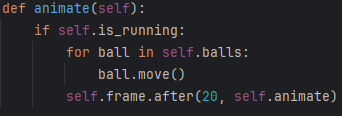
Bu fonksiyon, ekranda bir top oluşturup ekler. İlk olarak, ball = Ball(self.canvas, size, color, self.speed) ifadesi, Ball sınıfından yeni bir top nesnesi oluşturur. Bu nesne, self.canvas üzerine, verilen size (boyut), color (renk) ve self.speed (hız) parametreleriyle yaratılır. Sonrasında, self.balls.append(ball) ifadesi, yeni oluşturulan top nesnesini self.balls listesine ekler. Bu liste, animasyon sırasında hareket ettirilecek topların nesnelerini saklar.

**def take\_color(self, color):**

Bu fonksiyon, seçilen rengi günceller. self.selected\_color = color ifadesi, parametre olarak gelen color değerini self.selected\_color özelliğine atar. Bu sayede, kullanıcı bir renk butonuna tıkladığında, o rengin değeri self.selected\_color olarak kaydedilir ve top eklerken bu renk kullanılabilir.

**def start\_animation(self):**

Bu fonksiyon, animasyonu başlatmak için kullanılır. İlk olarak, if not self.is\_running: ifadesiyle animasyonun zaten çalışıp çalışmadığı kontrol edilir. Eğer animasyon başlamamışsa (self.is\_running False ise), self.is\_running = True ile animasyonun başladığı belirtir ve ardından self.animate() fonksiyonu çağırır. Bu, animasyonun hareket etmeye başlamasını sağlar.

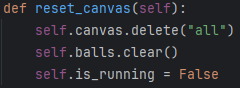
**def animate(self):**

Bu fonksiyon, animasyonu sürekli olarak günceller. İlk olarak, if self.is\_running: ifadesiyle animasyonun çalışıp çalışmadığı kontrol edilir. Eğer animasyon aktifse (self.is\_running True ise), for ball in self.balls: döngüsü ile ekranda bulunan her topa (self.balls listesindeki her bir topa) erişilir ve her top için ball.move() fonksiyonu çağrılır, bu da topların hareket etmesini sağlar. Son olarak, self.frame.after(20, self.animate) ifadesi ile 20 milisaniye sonra aynı fonksiyon (animate) tekrar çağrılır, böylece animasyon sürekli olarak güncellenir.

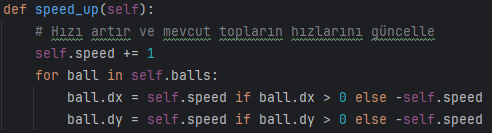
**def stop\_animation(self):**

Bu fonksiyon, animasyonu durdurmak için kullanılır. self.is\_running = False ifadesi, self.is\_running bayrağını False olarak ayarlar, böylece animasyon durur. Bu, animasyonun devam etmesini engeller ve hareket eden topların durmasını sağlar.

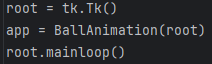
**def reset\_canvas(self):**

Bu fonksiyon, animasyonu sıfırlamak ve ekranı temizlemek için kullanılır. İlk olarak, self.canvas.delete("all") ifadesi, Canvas üzerindeki tüm öğeleri siler, yani ekrandaki tüm topları temizler. Sonra, self.balls.clear() ifadesi, topların saklandığı self.balls listesini boşaltır, yani topların nesnelerini siler. Son olarak, self.is\_running = False ifadesi, animasyonun durmasını sağlar ve animasyonun durduğunu belirtir.

**def speed\_up(self):**

Bu fonksiyon, animasyondaki topların hızını artırır. İlk olarak, self.speed += 1 ile hız değeri 1 artırılır. Sonrasında, for ball in self.balls: döngüsüyle tüm toplar üzerinden geçilir. Her bir topun yatay (ball.dx) ve dikey (ball.dy) hızları, ball.dx = self.speed if ball.dx > 0 else -self.speed ve ball.dy = self.speed if ball.dy > 0 else -self.speed ifadeleriyle güncellenir. Bu güncelleme, mevcut hız pozitifse hızın aynı yönde artırılmasını, negatifse ters yönde yapılmasını sağlar.

**Tkinder çalıştırma:**

tkinter uygulamasını başlatır. root = tk.Tk() ana pencereyi oluşturur. app = BallAnimation(root) ile BallAnimation sınıfından bir nesne yaratılır ve ana pencereye bağlanır. root.mainloop() ise uygulamanın ana döngüsünü başlatır, böylece kullanıcı etkileşimleri işlenir ve uygulama çalışmaya devam eder.

# SONUÇ VE ÖĞRENİLEN DERSLER

 **Tkinter Kullanımı**: Tkinter modülünün, Python ile GUI (Grafiksel Kullanıcı Arayüzü) oluşturmak için nasıl kullanıldığını öğrendik. Tk() sınıfı ile ana pencere oluşturulup, Canvas, Button gibi widget'lar kullanılarak arayüz elemanlarını oluşturduk.

 **Sınıf ve Nesne Kavramı**: Python'da sınıfların nasıl tanımlandığını ve nesne tabanlı programlamanın nasıl çalıştığını öğrendik. Ball ve BallAnimation sınıflarının işlevlerini ve birbirleriyle olan ilişkilerini inceledik.

 **Animasyon Mantığı**: Animasyon oluşturmanın temel mantığını öğrendik. Topların hareket etmesi için move() fonksiyonunun nasıl çalıştığını, animasyonu başlatmak, durdurmak ve hızlandırmak için kullanılan fonksiyonları inceledik.

 **Event Handling**: Kullanıcı etkileşimlerini (örneğin, butonlara tıklama) nasıl yakalayarak uygulamanın davranışını değiştirebileceğinizi öğrendik. bind() ve command özelliklerinin nasıl çalıştığını anlamış olduk.

 **Tkinter Döngüsü ve mainloop()**: Tkinter'ın ana döngüsünün (mainloop()) nasıl çalıştığını ve uygulamanın kullanıcı etkileşimlerine nasıl cevap verdiğini öğrendik.

# GİTHUB LİNKLERİ

Talha Tuna = <https://github.com/TalhaTuna2>

Emrah Şahin = <https://github.com/emrahsahn>