

T.C.

BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

OYUN PROGRAMLAMA

VİZE PROJE ÖDEVİ

SIRRI DEMİRTAŞ 2111502222

AYŞENUR ADA 2111502252

FEYZA KÖKSAL 191522200

**VİZE PROJE ÖDEVİ**

War of red and blue adında bir oyun geliştirildi. Oyuncular, kendi kulelerini korumak için "Kırmızı Asker" ve "Mavi Asker" karakterlerini kontrol ederler ve birbirlerine mermi fırlatarak rakibin canını azaltmaya çalışırlar. Oyunun temel işleyişi, askerlerin hareketleri, mermi atışları ve canının izlenmesini içeren iki kişilik aksiyon oyundur.

Programın içeriği ve fonksiyonları

1. Pygame modüllerinin başlatılması

Yazı tipi ve ses modülü başlatıldı.

pygame.font.init()

pygame.mixer.init()

1. Ekran ayarları

Pencere boyutları tanımlandı. Oyun penceresi oluşturuldu. Pencere başlıkları ayarlandı.

WIDTH, HEIGHT = 1100, 600

WIN = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("War of Red and Blue")

1. Renklerin tanımlanması

Oyuncu askerlerin renkleri red ve blue ,Metinler için white arka plan ve diğer kullanımlar için ise black olarak tanımlandı.

RED = (255, 0, 0)

BLUE = (0, 0, 255)

#  Diğer renklerle kontrast oluşturuyor: RED üzerine yazı yazarken vb.

WHITE = (255, 255, 255)

# Rengin henüz RED ya da BLUE olmadığı durumda kullanılır

BLACK = (0, 0, 0)

1. Fontlar

Askerin sağlığı için ve kazanan mesajı için fontlar tanımlandı.

HEALTH\_FONT = pygame.font.SysFont('verdana', 24)

WINNER\_FONT = pygame.font.SysFont('verdana', 48)

1. Oyundaki sabit ayarlar

# FPS

FPS = 60

# Asker ve mermi hızları

VELOCITY = 5

BULLET\_VELOCITY = 7

# Aynı anda atılabilecek mermi sayısı

MAX\_BULLETS = 5

# Asker boyutları

SOLDIER\_WIDTH = 55

SOLDIER\_HEIGHT = 60

1. Olaylar

Kırmızı askerin vurulduğu olay ve mavi askerin vurulduğu olaylar eklendi.

# Olaylar

RED\_HIT = pygame.USEREVENT + 1

BLUE\_HIT = pygame.USEREVENT + 2

1. Görsellerin yüklenmesi

Asker görsellerini yüklendi ve ölçeklendirildi. Arka plan görselini yüklendi ve ölçeklendirildi. Oyun simgesini yüklendi ve ayarlandı.

# Kırmızı asker görseli

RED\_SOLDIER\_IMAGE = pygame.image.load(

    os.path.join('assets', 'red\_soldier.png'))

RED\_SOLDIER = pygame.transform.rotate(pygame.transform.scale(

    RED\_SOLDIER\_IMAGE, (SOLDIER\_WIDTH, SOLDIER\_HEIGHT)), 0)

# Mavi asker görseli

BLUE\_SOLDIER\_IMAGE = pygame.image.load(

    os.path.join('assets', 'blue\_soldier.png'))

BLUE\_SOLDIER = pygame.transform.rotate(pygame.transform.scale(

    BLUE\_SOLDIER\_IMAGE, (SOLDIER\_WIDTH, SOLDIER\_HEIGHT)), 0)

# Arkaplan görseli

BACKGROUND = pygame.transform.scale(pygame.image.load(

    os.path.join('assets', 'background.png')), (WIDTH, HEIGHT))

# Oyun simgesi

ICON = pygame.image.load(os.path.join('assets', 'game\_icon.png'))

pygame.display.set\_icon(ICON)

1. Ses efekti Mermi ateşleme sesi yüklendi.

BULLET\_FIRE\_SOUND = pygame.mixer.Sound(

    os.path.join('assets', 'bullet\_sound.mp3'))

1. Soldier class

Oyun içindeki asker karakterlerinin özelliklerini ve hareketlerini yönetmek için tasarlandı. Bu class askerin pozisyonunu takip eder, ekranda görsellerini çizer, hareket etmelerini sağlar ve mermi işlemlerini yönetir.

class Soldier:

    '''

    '''

    def \_\_init\_\_(self, x, y, width, height, image):

        '''

        x: askerin x koordinatı

        y: askerin y koordinatı

        width: askerin genişliği

        height: askerin yüksekliği

        image: pygame.Surface, askerin görseli

        '''

        self.rect = pygame.Rect(x, y, width, height)

        self.image = image

        self.bullets = []

        self.health = 10

10) İnit metodu; asker nesnesinin başlangıç komutunu ayarlar.

self.rect: Askere ait dikdörtgen alanı belirtir ve pozisyonunu yönetir.

self.image: Askere ait görsel

self.bullets: Askere ait mermilerin listesi

self.health: Askere ait can puanı olarak tanımlandı.

11) Draw metodu

Draw metodu askeri ve ona ait olan mermileri ekrana çizdirdi. Merminin rengi askerin rengine göre belirlendi.

 def draw(self, window):

        window.blit(self.image, (self.rect.x, self.rect.y))

        for bullet in self.bullets:

          if self.image == RED\_SOLDIER:

                pygame.draw.rect(window, RED, bullet)

            else:

                pygame.draw.rect(window, BLUE, bullet)

Askere ait görsel (self.image) ve mermiler (self.bullets) ekrana çizildi.

12) Move metodu

Move metodu askerin klavye girişlerine göre hareket etmesini sağlar. Sınırların dışına çıkmaması için kontrol yapılır.

def move(self, keys, left, right, up, down, border\_limit):

    if keys[left]:  # Sol tuşa basılmışsa

            if self.rect.x - VELOCITY > border\_limit[0]:  # Sınırı geçmiyorsa

                self.rect.x -= VELOCITY

        if keys[right]:  # Sağ tuşa basılmışsa

            # Sınırı geçmiyorsa

            if self.rect.x + VELOCITY + self.rect.width < border\_limit[1]:

                self.rect.x += VELOCITY

        if keys[up]:  # Yukarı tuşa basılmışsa

            if self.rect.y - VELOCITY > 0:  # Sınırı geçmiyorsa

                self.rect.y -= VELOCITY

        if keys[down]:  # Aşağı tuşa basılmışsa

            # Sınırı geçmiyorsa

            if self.rect.y + VELOCITY + self.rect.height < HEIGHT - 15:

                self.rect.y += VELOCITY

keys: Tuşları ve basılı olup olmadıklarını tutar.

left, right, up, down: Hareket tuşları.

border\_limit: Askerin hareket edebileceği sınırlar olarak tanımlanmıştır.

13) handle\_bullets Metodu

Bu metod, askerin ateşlediği mermileri hareket ettirir ve diğer askere çarpıp çarpmadığını kontrol eder.

def handle\_bullets(self, other):

        for bullet in self.bullets:

            # Mermiyi hareket ettir

            if self.image == RED\_SOLDIER:

                # Kırmızı asker ise sağa hareket etsin

                bullet.x += BULLET\_VELOCITY

            else:

                # Mavi asker ise sola hareket etsin

                bullet.x -= BULLET\_VELOCITY

            # Mermi diğer askere çarparsa olayı tetikle

            if other.rect.colliderect(bullet):

                if self.image == BLUE\_SOLDIER:

                    pygame.event.post(pygame.event.Event(RED\_HIT))

                else:

                    pygame.event.post(pygame.event.Event(BLUE\_HIT))

                self.bullets.remove(bullet)

            # Mermi ekran dışına çıkarsa sil

            elif bullet.x > WIDTH or bullet.x < 0:

                self.bullets.remove(bullet)

Mermi diğer askere çarpıp çarpmadığı kontrol edilir. Eğer çarparsa, ilgili olay tetiklenir ve mermi listeden kaldırılır. Eğer ekran dışına çıkarsa, mermi yine listeden kaldırılır şekilde ayarlandı.

14) draw\_window(red, blue) fonksiyon

Bu fonksiyon oyunun ana oyun penceresini çizer. Fonksiyon, oyun alanının arka planını çizer, her iki takımın askerlerinin sağlık durumunu gösteren metinleri, yani kırmızı ve mavi askerlerin canlarını ekrana yazar, ardından kırmızı ve mavi askerleri çizer. Bu fonksiyon, kırmızı ve mavi askerlerin sağlık bilgilerini içeren red ve blue nesnelerini parametre olarak alır.

def draw\_window(red, blue):

    WIN.blit(BACKGROUND, (0, 0))

    red\_health\_text = HEALTH\_FONT.render("Health: " + str(red.health), 1, RED)

    blue\_health\_text = HEALTH\_FONT.render(

        "Health: " + str(blue.health), 1, BLUE)

    WIN.blit(blue\_health\_text, (WIDTH - blue\_health\_text.get\_width() - 10, 10))

    WIN.blit(red\_health\_text, (10, 10))

    red.draw(WIN)

    blue.draw(WIN)

    pygame.display.update()

15) draw\_winner(text, color) fonksiyon

Bu fonksiyon, oyunun bir galibini veya beraberliği belirten metni ekrana yazar. Fonksiyon, ekrana yazılacak metni ve metnin rengini parametre olarak alır. Metin, oyunun bitişini belirten bir mesaj olabilir ve genellikle oyunun sonunda çağrılır. Bu fonksiyon, metni ekrana yazmadan önce metnin boyutlarına uygun bir arka plan dikdörtgeni çizer ve ardından metni bu arka plan üzerine yazar. Fonksiyon 3 saniye bekler ve sonra ekranda gösterilen sonucun daha net görülmesini sağlamak için ekrandaki her şeyi günceller.

ef draw\_winner(text, color):

    draw\_text = WINNER\_FONT.render(text, 1, WHITE)

    background\_rect = pygame.Rect(WIDTH/2 - draw\_text.get\_width() / 2 - 20, HEIGHT/2 -

                                  draw\_text.get\_height()/2 - 20, draw\_text.get\_width() + 40, draw\_text.get\_height() + 40)

    pygame.draw.rect(WIN, color, background\_rect)

    WIN.blit(draw\_text, (WIDTH/2 - draw\_text.get\_width() /

             2, HEIGHT/2 - draw\_text.get\_height()/2))

    pygame.display.update()

    pygame.time.delay(3000)

Her iki fonksiyon da pygame.display.update() kullanarak ekranı günceller ve değişiklikleri gösterir.

16 main() fonksiyonu

Bu fonksiyon, oyunun ana döngüsünü başlatır. İki asker nesnesi oluşturur (kırmızı ve mavi asker), bir oyun saati (clock) başlatır ve oyun döngüsünü başlatır. Oyun döngüsü, kullanıcının çıkış yapana kadar çalışır (run True olduğu sürece). Her turda, oyun saniyede belirli bir kare hızında çalışır. askerlerin hareketini ve ateş etme yeteneklerini kontrol eder, çarpışma olaylarını kontrol eder (kırmızı veya mavi askerin vurulması durumunda), kazananı belirler ve ekrana çizim yapar.

def main():

    red = Soldier(200, 420, SOLDIER\_WIDTH, SOLDIER\_HEIGHT, RED\_SOLDIER)

    blue = Soldier(840, 425, SOLDIER\_WIDTH, SOLDIER\_HEIGHT, BLUE\_SOLDIER)

    clock = pygame.time.Clock()

    run = True

    while run:

        clock.tick(FPS)

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                run = False

                pygame.quit()

17) Ana döngü

Oyun sürekli olarak çalışırken (while run:) devam eder. Her çerçeve (frame) için belirlenen FPS (çerçeve başına saniye) değeriyle zaman kontrol edilir.

18) Olay işleme

pygame.event.get() ile her bir olay (event) işlenir.

Kullanıcının olaylarına göre (mermi ateşleme, tuş vuruşları) gibi işlemler yapılır.

for event in pygame.event.get():

            if event.type == pygame.QUIT:

                run = False

                pygame.quit()

19) Tuş vuruşları ve çarpışmanın işlenmesi

Kullanıcının klavyeden yaptığı tuş vuruşlarına göre, askerlerin mermileri ateşlenir (pygame.KEYDOWN olayı ile tuş vuruşları algılanır).

Her iki askerin maksimum mermi sayısını aşmaması için kontrol yapılır.

Askerlerin ve mermilerin çarpışmaları işlenir. Bir askerin vurulması durumunda can seviyesi düşürülür (RED\_HIT veya BLUE\_HIT olayları ile kontrol edilir).

 if event.type == pygame.KEYDOWN:

                # Farklı sistemler için (Windows, Mac, Linux)

                # farklı tuşlar kullanılabilir:

                # K\_LCTRL: sol CTRL tuşu, K\_LMETA: Mac command tuşu gibi

                # Kırmızı asker mermi ateşler.

                if event.key in [pygame.K\_LCTRL, pygame.K\_LMETA] and len(red.bullets) < MAX\_BULLETS:

                    BULLET\_FIRE\_SOUND.play()

                    bullet = pygame.Rect(

                        red.rect.x + red.rect.width, red.rect.y + red.rect.height // 2 - 2, 10, 5)

                    red.bullets.append(bullet)

                # Mavi asker mermi ateşler

                if event.key in [pygame.K\_RCTRL, pygame.K\_RMETA] and len(blue.bullets) < MAX\_BULLETS:

                    BULLET\_FIRE\_SOUND.play()

                    bullet = pygame.Rect(

                        blue.rect.x, blue.rect.y + blue.rect.height // 2 - 2, 10, 5)

                    blue.bullets.append(bullet)

            # Kırmızı asker vurulduysa

            if event.type == RED\_HIT:

                # Kırmızı askerin canını azalt

                red.health -= 1

            # Mavi asker vurulduysa

            if event.type == BLUE\_HIT:

                # Mavi askerin canını azalt

                blue.health -= 1

20) Oyun sonu ve kazanan belirleme koşulları

draw\_window() fonksiyonu çağrılarak oyun penceresi güncellenir ve çizilir.

Her çerçevede, askerlerin sağlık durumu kontrol edilerek bir kazanan belirlenir.

Eğer bir askerin sağlığı sıfıra düşerse, diğer asker oyunu kazanmış olur ve ilgili metin ekrana yazdırılır.

# Kazananı belirleme

        winner\_text = ""

        winner\_color = BLACK

        # Kırmızı asker kazanırsa

        if blue.health <= 0:

            winner\_text = "Red Soldier Wins!"

            winner\_color = RED

        # Mavi asker kazanırsa

        elif red.health <= 0:

            winner\_text = "Blue Soldier Wins!"

            winner\_color = BLUE

        # Kazanan varsa

        if winner\_text != "":

            draw\_winner(winner\_text, winner\_color)  # Kazananı göster

            break  # Oyunu bitir

   main()  # Oyunu bitince tekrar başlat

main() fonksiyonu, oyunun tüm işleyişini kontrol eder ve diğer fonksiyonları çağırarak oyunun doğru şekilde çalışmasını sağlar.

**EKLER;**

import pygame

import os

# Fontları kullanmak için gerekli

pygame.font.init()

# Ses eklemek için gerekli

pygame.mixer.init()

# Ekran ayarları

WIDTH, HEIGHT = 1100, 600

# Pencere oluştur

WIN = pygame.display.set\_mode((WIDTH, HEIGHT))

# Pencere başlığı

pygame.display.set\_caption("War of Red and Blue")

# RENKLER

# Askerlerle ilişiği olan renkler

RED = (255, 0, 0)

BLUE = (0, 0, 255)

#  Diğer renklerle kontrast oluşturuyor: RED üzerine yazı yazarken vb.

WHITE = (255, 255, 255)

# Rengin henüz RED ya da BLUE olmadığı durumda kullanılır

BLACK = (0, 0, 0)

# Fontlar

HEALTH\_FONT = pygame.font.SysFont('verdana', 24)

WINNER\_FONT = pygame.font.SysFont('verdana', 48)

# FPS

FPS = 60

# Asker ve mermi hızları

VELOCITY = 5

BULLET\_VELOCITY = 7

# Aynı anda atılabilecek mermi sayısı

MAX\_BULLETS = 5

# Asker boyutları

SOLDIER\_WIDTH = 55

SOLDIER\_HEIGHT = 60

# Olaylar

RED\_HIT = pygame.USEREVENT + 1

BLUE\_HIT = pygame.USEREVENT + 2

# Kırmızı asker görseli

RED\_SOLDIER\_IMAGE = pygame.image.load(

    os.path.join('assets', 'red\_soldier.png'))

RED\_SOLDIER = pygame.transform.rotate(pygame.transform.scale(

    RED\_SOLDIER\_IMAGE, (SOLDIER\_WIDTH, SOLDIER\_HEIGHT)), 0)

# Mavi asker görseli

BLUE\_SOLDIER\_IMAGE = pygame.image.load(

    os.path.join('assets', 'blue\_soldier.png'))

BLUE\_SOLDIER = pygame.transform.rotate(pygame.transform.scale(

    BLUE\_SOLDIER\_IMAGE, (SOLDIER\_WIDTH, SOLDIER\_HEIGHT)), 0)

# Arkaplan görseli

BACKGROUND = pygame.transform.scale(pygame.image.load(

    os.path.join('assets', 'background.png')), (WIDTH, HEIGHT))

# Oyun simgesi

ICON = pygame.image.load(os.path.join('assets', 'game\_icon.png'))

pygame.display.set\_icon(ICON)

# Mermi sesi

BULLET\_FIRE\_SOUND = pygame.mixer.Sound(

    os.path.join('assets', 'bullet\_sound.mp3'))

class Soldier:

'''

Bu class, askerlerin özelliklerini ve hareketlerini tanımlar.

'''

def \_\_init\_\_(self, x, y, width, height, image):

'''

x: askerin x koordinatı

y: askerin y koordinatı

width: askerin genişliği

height: askerin yüksekliği

image: pygame.Surface, askerin görseli

'''

self.rect = pygame.Rect(x, y, width, height)

self.image = image

self.bullets = []

self.health = 10

def draw(self, window):

'''

Askeri ve mermileri ekrana çizer.

'''

window.blit(self.image, (self.rect.x, self.rect.y))

for bullet in self.bullets:

# Eğer asker kırmızıysa mermi kırmızı olsun, değilse mavi

if self.image == RED\_SOLDIER:

pygame.draw.rect(window, RED, bullet)

else:

pygame.draw.rect(window, BLUE, bullet)

def move(self, keys, left, right, up, down, border\_limit):

'''

Askeri hareket ettirir.

keys: tuşları ve basılı olup olmadıklarını tutar

left: sola gitmek için atanmış tuş

right: sağa gitmek için atanmış tuş

up: yukarı gitmek için atanmış tuş

down: aşağı gitmek için atanmış tuş

border\_limit: askerin hareket edebileceği sınırlar

'''

if keys[left]: # Sol tuşa basılmışsa

if self.rect.x - VELOCITY > border\_limit[0]: # Sınırı geçmiyorsa

self.rect.x -= VELOCITY

if keys[right]: # Sağ tuşa basılmışsa

# Sınırı geçmiyorsa

if self.rect.x + VELOCITY + self.rect.width < border\_limit[1]:

self.rect.x += VELOCITY

if keys[up]: # Yukarı tuşa basılmışsa

if self.rect.y - VELOCITY > 0: # Sınırı geçmiyorsa

self.rect.y -= VELOCITY

if keys[down]: # Aşağı tuşa basılmışsa

# Sınırı geçmiyorsa

if self.rect.y + VELOCITY + self.rect.height < HEIGHT - 15:

self.rect.y += VELOCITY

def handle\_bullets(self, other):

'''

Mermileri hareket ettirir ve diğer askere çarptığında olayı tetikler.

'''

for bullet in self.bullets:

# Mermiyi hareket ettir

if self.image == RED\_SOLDIER:

# Kırmızı asker ise sağa hareket etsin

bullet.x += BULLET\_VELOCITY

else:

# Mavi asker ise sola hareket etsin

bullet.x -= BULLET\_VELOCITY

# Mermi diğer askere çarparsa olayı tetikle

if other.rect.colliderect(bullet):

if self.image == BLUE\_SOLDIER:

pygame.event.post(pygame.event.Event(RED\_HIT))

else:

pygame.event.post(pygame.event.Event(BLUE\_HIT))

self.bullets.remove(bullet)

# Mermi ekran dışına çıkarsa sil

elif bullet.x > WIDTH or bullet.x < 0:

self.bullets.remove(bullet)

ef draw\_window(red, blue):

'''

Pencereyi, skoru ve askerleri çizer.

'''

WIN.blit(BACKGROUND, (0, 0)) # Arkaplanı çiz

# Üst köşelerde kırmızı ve mavi askerlerin canını göster

red\_health\_text = HEALTH\_FONT.render("Health: " + str(red.health), 1, RED)

blue\_health\_text = HEALTH\_FONT.render(

"Health: " + str(blue.health), 1, BLUE)

WIN.blit(blue\_health\_text, (WIDTH - blue\_health\_text.get\_width() - 10, 10))

WIN.blit(red\_health\_text, (10, 10))

red.draw(WIN)

blue.draw(WIN)

pygame.display.update()

def draw\_winner(text, color):

draw\_text = WINNER\_FONT.render(text, 1, WHITE)

background\_rect = pygame.Rect(WIDTH/2 - draw\_text.get\_width() / 2 - 20, HEIGHT/2 -

draw\_text.get\_height()/2 - 20, draw\_text.get\_width() + 40, draw\_text.get\_height() + 40)

pygame.draw.rect(WIN, color, background\_rect)

WIN.blit(draw\_text, (WIDTH/2 - draw\_text.get\_width() /

2, HEIGHT/2 - draw\_text.get\_height()/2))

pygame.display.update()

pygame.time.delay(3000) # 3 saniye bekle

def main():

# Kırmızı ve mavi askerleri oluştur

red = Soldier(200, 420, SOLDIER\_WIDTH, SOLDIER\_HEIGHT, RED\_SOLDIER)

blue = Soldier(840, 425, SOLDIER\_WIDTH, SOLDIER\_HEIGHT, BLUE\_SOLDIER)

clock = pygame.time.Clock()

run = True

while run:

clock.tick(FPS)

for event in pygame.event.get():

if event.type == pygame.QUIT:

run = False

pygame.quit()

if event.type == pygame.KEYDOWN:

# Farklı sistemler için (Windows, Mac, Linux)

# farklı tuşlar kullanılabilir:

# K\_LCTRL: sol CTRL tuşu, K\_LMETA: Mac command tuşu gibi

# Kırmızı asker mermi ateşler.

if event.key in [pygame.K\_LCTRL, pygame.K\_LMETA] and len(red.bullets) < MAX\_BULLETS:

BULLET\_FIRE\_SOUND.play()

bullet = pygame.Rect(

red.rect.x + red.rect.width, red.rect.y + red.rect.height // 2 - 2, 10, 5)

red.bullets.append(bullet)

# Mavi asker mermi ateşler

if event.key in [pygame.K\_RCTRL, pygame.K\_RMETA] and len(blue.bullets) < MAX\_BULLETS:

BULLET\_FIRE\_SOUND.play()

bullet = pygame.Rect(

blue.rect.x, blue.rect.y + blue.rect.height // 2 - 2, 10, 5)

blue.bullets.append(bullet)

# Kırmızı asker vurulduysa

if event.type == RED\_HIT:

# Kırmızı askerin canını azalt

red.health -= 1

# Mavi asker vurulduysa

if event.type == BLUE\_HIT:

# Mavi askerin canını azalt

blue.health -= 1

# Kazananı belirleme

winner\_text = ""

winner\_color = BLACK

# Kırmızı asker kazanırsa

if blue.health <= 0:

winner\_text = "Red Soldier Wins!"

winner\_color = RED

# Mavi asker kazanırsa

elif red.health <= 0:

winner\_text = "Blue Soldier Wins!"

winner\_color = BLUE

# Kazanan varsa

if winner\_text != "":

draw\_winner(winner\_text, winner\_color) # Kazananı göster

break # Oyunu bitir

# Basılı tuşları al

keys\_pressed = pygame.key.get\_pressed()

# Kırmızı askeri hareket ettir

red.move(keys\_pressed, pygame.K\_a, pygame.K\_d,

pygame.K\_w, pygame.K\_s, (0, WIDTH // 2 - 5))

# Mavi askeri hareket ettir

blue.move(keys\_pressed, pygame.K\_LEFT, pygame.K\_RIGHT,

pygame.K\_UP, pygame.K\_DOWN, (WIDTH // 2 + 5, WIDTH))

# Kırmızı askerin mermilerini yönet

red.handle\_bullets(blue)

# Mavi askerin mermilerini yönet

blue.handle\_bullets(red)

# Ekranı güncelle

draw\_window(red, blue)

main() # Oyunu bitince tekrar başlat