import pygame  
from random import randrange  
  
RES = 600  
SIZE = 30  
  
x, y = 150, 150 # randrange(0, RES, SIZE), randrange(0, RES, SIZE) # Случайные значения для стартовой позиции  
apple = randrange(0, RES, SIZE), randrange(0, RES, SIZE)  
apple2 = randrange(0, (RES - SIZE), SIZE), randrange(0, (RES - SIZE), SIZE)  
dirs = {'W': True, 'S': True, 'A': True, 'D': True}  
length = 1  
snake = [(x, y)]  
dx, dy = 0, 0 # Здесь можно задать сразу направление движения на старте, без кнопок -да єто смещение я понял  
score = 0  
fps = 5  
  
pygame.init()  
sc = pygame.display.set\_mode([RES, RES])  
  
clock = pygame.time.Clock()  
font\_score = pygame.font.SysFont('Arial', 26, bold=True)  
font\_end = pygame.font.SysFont('Arial', 66, bold=True)  
img = pygame.image.load('assets/bg.png').convert() # рисунок для фона  
  
while True:  
 sc.fill(pygame.Color('black')) # закрашивание фона  
 # sc.blit(img, 0, 0)  
 # рисуем змейку  
 [(pygame.draw.rect(sc, pygame.Color('Green'), (i, j, SIZE - 2, SIZE - 2))) for i, j in snake]  
 pygame.draw.rect(sc, pygame.Color('red'), (\*apple, SIZE, SIZE))  
 pygame.draw.rect(sc, pygame.Color('blue'), (\*apple2, SIZE, SIZE)) # apple  
 render\_score = font\_score.render(f'SCORE: {score}', True, pygame.Color('orange'))  
 sc.blit(render\_score, (5, 5))  
 # snake movement  
 x += dx \* SIZE  
 y += dy \* SIZE  
 # snake.append((x, y)) # *TODO: Check the double append* snake.append((x, y))  
 snake = snake[-length:]  
 # eating apple/2  
 if snake[-1] == apple: # Это называется "точки останова". Чтобы прямо посереди кода останавливать его и смотреть, что происходит  
 apple = randrange(0, RES, SIZE), randrange(0, RES, SIZE)  
 length += 1  
 score += 1  
 fps += 1  
 snake.append(apple) # Здесь добавляем в змейку координаты яблока  
 if snake[-1] == apple2:  
 apple2 = randrange(0, RES, SIZE), randrange(0, RES, SIZE)  
 length += 1  
 score += 1  
 fps -= 1  
 snake.append(apple2) # Здесь добавляем в змейку координаты яблока 2 - яэто понял - когда ты добавлять стал - что я не указал как себя вести  
   
  
 if x < 0 or x > RES - SIZE or y > RES - SIZE or y < 0: # вот тут повнимательней!!! len(snake) != len(set(snake))  
 # print('Game over')  
 while True:  
 render\_end = font\_end.render('GAME OVER', 1, pygame.Color('orange'))  
 sc.blit(render\_end, (RES // 2 - 200, RES // 3))  
 pygame.display.flip()  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 exit()  
  
 pygame.display.flip()  
 clock.tick(fps)  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 exit()  
  
 # control  
 key = pygame.key.get\_pressed()  
 if key[pygame.K\_w]:  
 print('Key pressed W')  
 if key[pygame.K\_s]:  
 print('Key pressed S')  
 if key[pygame.K\_a]:  
 print('Key pressed A')  
 if key[pygame.K\_d]:  
 print('Key pressed D')  
  
 if key[pygame.K\_w] and dirs['W']: # Если нажали W то не можем нажать S  
 dx, dy = 0, -1  
 dirs = {'W': True, 'S': False, 'A': True, 'D': True}  
 if key[pygame.K\_s] and dirs['S']: # Если нажали S то не можем нажать W  
 dx, dy = 0, 1  
 dirs = {'W': False, 'S': True, 'A': True, 'D': True}  
 if key[pygame.K\_a] and dirs['A']:  
 dx, dy = -1, 0  
 dirs = {'W': True, 'S': True, 'A': True, 'D': False}  
 if key[pygame.K\_d] and dirs['D']:  
 dx, dy = 1, 0  
 dirs = {'W': True, 'S': True, 'A': False, 'D': True}

2. переделанній вариант с N колл-вом яблок

import pygame  
from random import randrange  
  
RES = 600  
SIZE = 30  
apple\_count = 7  
  
# Получить новые координаты одного нового яблока  
def get\_apple():  
 return [randrange(0, RES, SIZE), randrange(0, RES, SIZE)] # Example: [15, 10]  
  
  
''' Approach 1  
def get\_apples(count):  
 result = []  
 for i in range(0, count):  
 result.append(get\_apple())  
 return result  
'''  
  
# Получить список координат для указанного количества яблок  
# \* Использует метод get\_apple выше  
# Отвечая на вопрос. Ну вот создается список яблок из количества  
# так указано было только по одной координате у яблок при присваивании  
def get\_apples(count):  
 return [get\_apple() for i in range(0, count)] # Example: [[15,10], [20,15]]  
  
  
x, y = 150, 150 # randrange(0, RES, SIZE), randrange(0, RES, SIZE) # Случайные значения для стартовой позиции  
# apple = randrange(0, RES, SIZE), randrange(0, RES, SIZE)  
  
# Теперь вместо этого, мы можем писать так  
# Потому что этот метод возвращает то же самое, что и здесь справа от знака равно  
# Здесь кортеж был указан "коротким способом" это когда нет скобок вокруг, но 2 значения через запятую  
# a = 1,2 это то же самое что и a = (1,2)  
#apple\_backup = (randrange(0, (RES - SIZE), SIZE), randrange(0, (RES - SIZE), SIZE))  
apple\_backup = get\_apple()  
  
# Первое: Меняем кортеж на список  
# Второе: Выносим в ф-цию т.к. она нам еще пригодится  
apples = get\_apples(apple\_count) # Ну другое же дело, и красивее выглядит и понятно  
  
dirs = {'W': True, 'S': True, 'A': True, 'D': True}  
length = 1  
snake = [[x, y]]  
dx, dy = 0, 0 # Здесь можно задать сразу направление движения на старте, без кнопок -да єто смещение я понял  
score = 0  
fps = 5  
  
  
pygame.init()  
sc = pygame.display.set\_mode([RES, RES])  
  
clock = pygame.time.Clock()  
font\_score = pygame.font.SysFont('Arial', 26, bold=True)  
font\_end = pygame.font.SysFont('Arial', 66, bold=True)  
background = pygame.image.load('assets/1.jpg').convert() # рисунок для фона  
  
def draw\_background():  
 # sc.fill(pygame.Color('black'))  
 sc.blit(background, (0, 0))  
  
def draw\_snake():  
 # рисуем змейку  
 [(pygame.draw.rect(sc, pygame.Color('Green'), (i, j, SIZE - 2, SIZE - 2))) for i, j in snake]  
  
def draw\_apples():  
 for apple in apples:  
 # То, о чем я говорил "взять вручную" координаты из яблока  
 pygame.draw.rect(sc, pygame.Color('red'), (apple[0], apple[1], SIZE, SIZE))  
 pygame.draw.rect(sc, pygame.Color('blue'), (\*apple\_backup, SIZE, SIZE)) # backup apple  
 print('apl 0:', apple[0],'apl 1:', apple[1])  
  
  
def draw\_score():  
 render\_score = font\_score.render(f'SCORE: {score}', True, pygame.Color('orange'))  
 sc.blit(render\_score, (5, 5))  
  
def get\_head():  
 global snake  
 return snake[-1][0], snake[-1][1]  
  
def check\_apple\_collision():  
 global apple\_backup  
 global length  
 global score  
 global fps  
  
 if snake[-1] == apple\_backup:  
 apple\_backup = get\_apple()  
 length += 1  
 score += 1  
 fps -= 1  
 snake.append(apple\_backup)  
 for apple in apples:  
 # Потому что snake в себе содержит список списков (кортежей, не важно). ((1,1),(2,2)) ...  
 # Берем последний, получается элемент (2,2) и из него по очереди берем по нулевому индексу и по первому  
 # Ну или можно вариант проще, в одну строку развертыванием. Чтобы тебя не запутывать, сделаю как ниже  
 # Здесь я просто сохраняю координаты в удобно читаемые названия переменных.  
 snake\_x, snake\_y = get\_head()  
 apple\_x = apple[0] # Здесь одна координата, правильно. Но мы то перебираем целый список apples.  
 apple\_y = apple[1]  
  
 # Это вроде очевидно максимально, что тут написано ниже-- да понял - ядумал просто что они не нужні в форе  
 if snake\_x == apple\_x and snake\_y == apple\_y:  
 length += 1  
 score += 1  
 fps += 1  
 snake.append([apple\_x, apple\_y]) # Здесь добавляем в змейку координаты яблока  
  
 # Здесь, если условие сработало, то мы этому яблоку берем новые координаты  
 # Технически, мы его даже не удаляем , а откуда мі знамем что оно[0,1]  
 new\_x, new\_y = get\_apple()  
 apple[0] = new\_x  
 apple[1] = new\_y  
  
def move\_snake():  
 global x  
 global y  
 global snake  
  
 x += dx \* SIZE  
 y += dy \* SIZE  
 snake.append((x, y))  
 snake = snake[-length:]  
  
 check\_game\_over(x, y)  
  
  
def check\_game\_over(x, y):  
 if x < 0 or x > RES - SIZE or y > RES - SIZE or y < 0: # вот тут повнимательней!!! len(snake) != len(set(snake))  
 while True:  
 render\_end = font\_end.render('GAME OVER', 1, pygame.Color('orange'))  
 sc.blit(render\_end, (RES // 2 - 200, RES // 3))  
 pygame.display.flip()  
 for e in pygame.event.get():  
 if e.type == pygame.QUIT:  
 exit()  
  
  
def control\_snake():  
 global dirs  
 global dx  
 global dy  
  
 key = pygame.key.get\_pressed()  
 if key[pygame.K\_w]:  
 print('Key pressed W')  
 if key[pygame.K\_s]:  
 print('Key pressed S')  
 if key[pygame.K\_a]:  
 print('Key pressed A')  
 if key[pygame.K\_d]:  
 print('Key pressed D')  
  
 if key[pygame.K\_w] and dirs['W']: # Если нажали W то не можем нажать S  
 dx, dy = 0, -1  
 dirs = {'W': True, 'S': False, 'A': True, 'D': True}  
 if key[pygame.K\_s] and dirs['S']: # Если нажали S то не можем нажать W  
 dx, dy = 0, 1  
 dirs = {'W': False, 'S': True, 'A': True, 'D': True}  
 if key[pygame.K\_a] and dirs['A']:  
 dx, dy = -1, 0  
 dirs = {'W': True, 'S': True, 'A': True, 'D': False}  
 if key[pygame.K\_d] and dirs['D']:  
 dx, dy = 1, 0  
 dirs = {'W': True, 'S': True, 'A': False, 'D': True}  
  
  
  
def quit\_listener():  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 exit()  
  
  
while True:  
 # За то, намного легче смотреть и находить что за что отвечает. А не всё в одной куче - ну давай  
 draw\_background()  
 draw\_snake()  
 draw\_apples()  
 draw\_score()  
  
 control\_snake()  
  
 check\_apple\_collision()  
 move\_snake()  
  
 pygame.display.flip()  
 clock.tick(fps)  
  
 quit\_listener()

3 .модификация – добавлены другой тип яблок – уменьшает скорость и размер змеи , добавлено условия если змея уменьшается меньше головы – гейм овер

import pygame  
from random import randrange  
  
RES = 600  
SIZE = 30  
apple\_count = 3  
apple\_count\_bl = 2  
  
# Получить новые координаты одного нового яблока  
def get\_apple():  
 return [randrange(0, RES, SIZE), randrange(0, RES, SIZE)] # Example: [15, 10]  
  
def get\_apple\_bl():  
 return [randrange(0, RES, SIZE), randrange(0, RES, SIZE)] # Example: [15, 10]  
  
''' Approach 1  
def get\_apples(count):  
 result = []  
 for i in range(0, count):  
 result.append(get\_apple())  
 return result  
'''  
  
# Получить список координат для указанного количества яблок  
# \* Использует метод get\_apple выше  
# Отвечая на вопрос. Ну вот создается список яблок из количества  
# так указано было только по одной координате у яблок при присваивании  
def get\_apples(count):  
 return [get\_apple() for i in range(0, count)] # Example: [[15,10], [20,15]]  
  
def get\_apples\_bl(count):  
 return [get\_apple\_bl() for i in range(0, count)] # Example: [[15,10], [20,15]]  
  
x, y = randrange(0, RES, SIZE), randrange(0, RES, SIZE) # Случайные значения для стартовой позиции  
# apple = randrange(0, RES, SIZE), randrange(0, RES, SIZE)  
  
# Теперь вместо этого, мы можем писать так  
# Потому что этот метод возвращает то же самое, что и здесь справа от знака равно  
# Здесь кортеж был указан "коротким способом" это когда нет скобок вокруг, но 2 значения через запятую  
# a = 1,2 это то же самое что и a = (1,2)  
#apple\_backup = (randrange(0, (RES - SIZE), SIZE), randrange(0, (RES - SIZE), SIZE))  
#apple\_backup = get\_apple()  
  
# Первое: Меняем кортеж на список  
# Второе: Выносим в ф-цию т.к. она нам еще пригодится  
apples = get\_apples(apple\_count) # Ну другое же дело, и красивее выглядит и понятно  
apples\_bl = get\_apples\_bl(apple\_count\_bl)  
  
dirs = {'W': True, 'S': True, 'A': True, 'D': True}  
length = 1  
snake = [[x, y]]  
dx, dy = 0, 0 # Здесь можно задать сразу направление движения на старте, без кнопок -да єто смещение я понял  
score = 0  
fps = 5  
  
pygame.init()  
sc = pygame.display.set\_mode([RES, RES])  
clock = pygame.time.Clock()  
font\_score = pygame.font.SysFont('Arial', 26, bold=True)  
font\_end = pygame.font.SysFont('Arial', 66, bold=True)  
#background = pygame.image.load('assets/1.jpg').convert() # рисунок для фона  
  
def draw\_background():  
 sc.fill(pygame.Color('black'))  
 #sc.blit(background, (0, 0))  
  
def draw\_snake():  
 # рисуем змейку  
 [(pygame.draw.rect(sc, pygame.Color('Green'), (i, j, SIZE - 2, SIZE - 2))) for i, j in snake]  
  
def draw\_apples():  
 for apple in apples:  
 # То, о чем я говорил "взять вручную" координаты из яблока  
 pygame.draw.rect(sc, pygame.Color('red'), (apple[0], apple[1], SIZE, SIZE))  
 #pygame.draw.rect(sc, pygame.Color('blue'), (apple\_bl[0], apple\_bl[1], SIZE, SIZE)) # backup apple  
 print('apl 0:', apple[0],'apl 1:', apple[1])  
  
def draw\_apples\_bl():  
 for apple\_bl in apples\_bl:  
 pygame.draw.rect(sc, pygame.Color('blue'), (apple\_bl[0], apple\_bl[1], SIZE, SIZE)) # backup apple  
 print('apl 0:', apple\_bl[0], 'apl 1:', apple\_bl[1])  
  
def draw\_score():  
 render\_score = font\_score.render(f'SCORE: {score}', True, pygame.Color('orange'))  
 sc.blit(render\_score, (5, 5))  
  
def get\_head():  
 global snake  
 return snake[-1][0], snake[-1][1]  
  
def check\_apple\_collision():  
 global length  
 global score  
 global fps  
  
 for apple in apples\_bl:  
 snake\_x, snake\_y = get\_head()  
 apple\_x = apple[0]  
 apple\_y = apple[1]  
  
 if snake\_x == apple\_x and snake\_y == apple\_y:  
 length -= 1  
 score += 1  
 fps -= 1  
 snake.append([apple\_x, apple\_y])  
 new\_x, new\_y = get\_apple()  
 apple[0] = new\_x  
 apple[1] = new\_y  
  
 for apple in apples:  
 # Потому что snake в себе содержит список списков (кортежей, не важно). ((1,1),(2,2)) ...  
 # Берем последний, получается элемент (2,2) и из него по очереди берем по нулевому индексу и по первому  
 # Ну или можно вариант проще, в одну строку развертыванием. Чтобы тебя не запутывать, сделаю как ниже  
 # Здесь я просто сохраняю координаты в удобно читаемые названия переменных.  
 snake\_x, snake\_y = get\_head()  
 apple\_x = apple[0] # Здесь одна координата, правильно. Но мы то перебираем целый список apples.  
 apple\_y = apple[1]  
  
 # Это вроде очевидно максимально, что тут написано ниже-- да понял - ядумал просто что они не нужні в форе  
 if snake\_x == apple\_x and snake\_y == apple\_y:  
 length += 1  
 score += 1  
 fps += 1  
 snake.append([apple\_x, apple\_y]) # Здесь добавляем в змейку координаты яблока  
  
 # Здесь, если условие сработало, то мы этому яблоку берем новые координаты  
 # Технически, мы его даже не удаляем , а откуда мі знамем что оно[0,1]  
 new\_x, new\_y = get\_apple()  
 apple[0] = new\_x  
 apple[1] = new\_y  
  
def move\_snake():  
 global x  
 global y  
 global snake  
  
 x += dx \* SIZE  
 y += dy \* SIZE  
 snake.append((x, y))  
 snake = snake[-length:]  
  
 check\_game\_over(x, y)  
  
  
def check\_game\_over(x, y):  
 global length  
 if x < 0 or x > RES - SIZE or y > RES - SIZE or y < 0 or length <1: # границы экрана  
 while True:  
 render\_end = font\_end.render('GAME OVER', 1, pygame.Color('orange'))  
 sc.blit(render\_end, (RES // 2 - 200, RES // 3))  
 pygame.display.flip()  
 for e in pygame.event.get():  
 if e.type == pygame.QUIT:  
 exit()  
  
def control\_snake():  
 global dirs  
 global dx  
 global dy  
  
 key = pygame.key.get\_pressed()  
 if key[pygame.K\_w]:  
 print('Key pressed W')  
 if key[pygame.K\_s]:  
 print('Key pressed S')  
 if key[pygame.K\_a]:  
 print('Key pressed A')  
 if key[pygame.K\_d]:  
 print('Key pressed D')  
  
 if key[pygame.K\_w] and dirs['W']: # Если нажали W то не можем нажать S  
 dx, dy = 0, -1  
 dirs = {'W': True, 'S': False, 'A': True, 'D': True}  
 if key[pygame.K\_s] and dirs['S']: # Если нажали S то не можем нажать W  
 dx, dy = 0, 1  
 dirs = {'W': False, 'S': True, 'A': True, 'D': True}  
 if key[pygame.K\_a] and dirs['A']:  
 dx, dy = -1, 0  
 dirs = {'W': True, 'S': True, 'A': True, 'D': False}  
 if key[pygame.K\_d] and dirs['D']:  
 dx, dy = 1, 0  
 dirs = {'W': True, 'S': True, 'A': False, 'D': True}  
  
def quit\_listener():  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 exit()  
  
while True:  
 # За то, намного легче смотреть и находить что за что отвечает. А не всё в одной куче - ну давай  
 draw\_background()  
 draw\_snake()  
 draw\_apples()  
 draw\_apples\_bl()  
 draw\_score()  
  
 control\_snake()  
  
 check\_apple\_collision()  
 move\_snake()  
  
 pygame.display.flip()  
 clock.tick(fps)  
  
 quit\_listener()

ошибки ( недочеты ) !!!

quit\_listener()  
Вот так делаешь и всё)  
apples = get\_apples(apple\_count)  
apples\_bl = get\_applesl(apple\_count\_bl)  
def draw\_apples(apples\_list, color):  
 for apple in apples\_list:  
 pygame.draw.rect(sc, pygame.Color(color), (apple[0], apple[1], SIZE, SIZE))  
И там где вызываешь, вместо двух разных методов вызываешь так:  
draw\_apples(apples, 'red')  
draw\_apples(apples\_bl, 'blue')