Ant Simulation with Food

Working:
This Python program simulates ants collecting food and bringing it back to a nest using Pygame for graphical visualization. In the simulation, ants perform random walks until they find food, pick it up, and return it to the nest, with collected food being removed from the map. Additional food can be spawned by clicking the mouse, and most food is placed near the nest for faster collection. The nest, food, and ants are color-coded (RED for the nest, GREEN for food, BLACK for ants), and a displays current simulation statistics. The program allows modification of the number of ants or food sources to observe different behaviors and dynamics.

Major Classes & Functions:

1.

```
class Nest(pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.image = pygame.Surface((24, 24))
        self.image.fill(RED)
        self.rect = self.image.get_rect(center=(x, y))
        self.food_collected = 0
```

This function shows the nest where ants deposit their food and tracks total food collected during the process.

2.

```
class Food(pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.image = pygame.Surface((8, 8))
        self.image.fill(GREEN)
        self.rect = self.image.get_rect(center=(x, y))
        self.collected = False
```

This function shows food scattered on the screen which can be collected by the ants.

3.

```
class Ant(pygame.sprite.Sprite):
    def __init__(self, x, y, nest):
        super().__init__()
        self.image = pygame.Surface((6, 6))
        self.image.fill(BLACK)
        self.rect = self.image.get_rect(center=(x, y))

        self.pos = pygame.math.Vector2(self.rect.center)
        self.speed = 1.8
        self.state = "searching"
        self.target_food = None
        self.nest = nest
```

This function represents ants moving which has two states 1. Searching and 2. returning

4.

```
    def update(self, food_group):
        if self.state == "searching":
            jitter = pygame.math.Vector2(random.uniform(-1.8, 1.8), random.uniform(-1.8, 1.8))
            self.pos += jitter

            self.pos.x = max(0, min(self.pos.x, SCREEN_WIDTH))
            self.pos.y = max(0, min(self.pos.y, SCREEN_HEIGHT))
            self.rect.center = (round(self.pos.x), round(self.pos.y))

            for food in list(food_group):
                if self.rect.colliderect(food.rect):
                    self.state = "returning"
                    self.target_food = food
                    food.collected = True
                    food_group.remove(food)
                    break

        elif self.state == "returning":
            nest_vec = pygame.math.Vector2(self.nest.rect.center)
            direction = (nest_vec - self.pos)
            distance = direction.length()
            if distance > 0:
                direction = direction.normalize()
                self.pos += direction * self.speed

            self.rect.center = (round(self.pos.x), round(self.pos.y))

            if self.rect.colliderect(self.nest.rect):
                self.nest.food_collected += 1
                self.state = "searching"
                self.target_food = None
                away = pygame.math.Vector2(random.uniform(-5, 5), random.uniform(-5, 5))
                self.pos += away
                self.rect.center = (round(self.pos.x), round(self.pos.y))
```

This function handles ant behavior, their collision with food and returning to the nest.