In [ ]:

```
import pygame, sys
import numpy as np
import random


BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
BLUE = (0, 100, 255)
GREEN = (255, 0, 0)
PURPLE = (50, 150, 50)
GREY = (230, 230, 230)
HORRIBLE_YELLOW = (190, 175, 50)

BACKGROUND = WHITE


class Dot(pygame.sprite.Sprite):
    def __init__(
        self,
        x,
        y,
        width,
        height,
        color=BLACK,
        radius=5,
        velocity=[0, 0],
        randomize=False,
    ):
        super().__init__()
        self.image = pygame.Surface([radius * 2, radius * 2])
        self.image.fill(BACKGROUND)
        pygame.draw.circle(
            self.image, color, (radius, radius), radius
        )

        self.rect = self.image.get_rect()
        self.pos = np.array([x, y], dtype=np.float64)
        self.vel = np.asarray(velocity, dtype=np.float64)

        self.killswitch_on = False
        self.recovered = False
        self.randomize = randomize

        self.WIDTH = width
        self.HEIGHT = height

    def update(self):

        self.pos += self.vel

        x, y = self.pos

        # Periodic boundary conditions
        if x < 0:
            self.pos[0] = self.WIDTH
            x = self.WIDTH
        if x > self.WIDTH:
            self.pos[0] = 0
            x = 0
        if y < 0:
            self.pos[1] = self.HEIGHT
            y = self.HEIGHT
        if y > self.HEIGHT:
            self.pos[1] = 0
            y = 0

        self.rect.x = x
        self.rect.y = y

        vel_norm = np.linalg.norm(self.vel)
        if vel_norm > 3:
            self.vel /= vel_norm

        if self.randomize:
            self.vel += np.random.rand(2) * 2 - 1

        if self.killswitch_on:
            self.cycles_to_fate -= 1

            if self.cycles_to_fate <= 0:
                self.killswitch_on = False
                some_number = np.random.rand()
                if self.mortality_rate > some_number:
                    self.kill()
                else:
                    self.recovered = True

    def respawn(self, color, radius=5):
        return Dot(
            self.rect.x,
            self.rect.y,
            self.WIDTH,
            self.HEIGHT,
            color=color,
            velocity=self.vel,
        )

    def killswitch(self, cycles_to_fate=20, mortality_rate=0.2):
        self.killswitch_on = True
        self.cycles_to_fate = cycles_to_fate
        self.mortality_rate = mortality_rate


class Simulation:
    def __init__(self, width=600, height=480):
        self.WIDTH = width
        self.HEIGHT = height

        self.susceptible_container = pygame.sprite.Group()
        self.infected_container = pygame.sprite.Group()
        self.recovered_container = pygame.sprite.Group()
        self.all_container = pygame.sprite.Group()

        self.n_susceptible = 20
        self.n_infected = 1
        self.n_quarantined = 0
        self.T = 1000
        self.cycles_to_fate = 20
        self.mortality_rate = 0.2

    def start(self, randomize=False):

        self.N = (
            self.n_susceptible + self.n_infected + self.n_quarantined
        )

        pygame.init()
        screen = pygame.display.set_mode([self.WIDTH, self.HEIGHT])

        for i in range(self.n_susceptible):
            x = np.random.randint(0, self.WIDTH + 1)
            y = np.random.randint(0, self.HEIGHT + 1)
            vel = np.random.rand(2) * 2 - 1
            guy = Dot(
                x,
                y,
                self.WIDTH,
                self.HEIGHT,
                color=BLUE,
                velocity=vel,
                randomize=randomize,
            )
            self.susceptible_container.add(guy)
            self.all_container.add(guy)

        for i in range(self.n_quarantined):
            x = np.random.randint(0, self.WIDTH + 1)
            y = np.random.randint(0, self.HEIGHT + 1)
            vel = [0, 0]
            guy = Dot(
                x,
                y,
                self.WIDTH,
                self.HEIGHT,
                color=BLUE,
                velocity=vel,
                randomize=False,
            )
            self.susceptible_container.add(guy)
            self.all_container.add(guy)

        for i in range(self.n_infected):
            x = np.random.randint(0, self.WIDTH + 1)
            y = np.random.randint(0, self.HEIGHT + 1)
            vel = np.random.rand(2) * 2 - 1
            guy = Dot(
                x,
                y,
                self.WIDTH,
                self.HEIGHT,
                color=GREEN,
                velocity=vel,
                randomize=randomize,
            )
            self.infected_container.add(guy)
            self.all_container.add(guy)

        stats = pygame.Surface((self.WIDTH // 4, self.HEIGHT // 4))
        stats.fill(GREY)
        stats.set_alpha(230)
        stats_pos = (self.WIDTH // 40, self.HEIGHT // 40)

        clock = pygame.time.Clock()

        for i in range(self.T):
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    sys.exit()

            self.all_container.update()

            screen.fill(BACKGROUND)

            # Update stats
            stats_height = stats.get_height()
            stats_width = stats.get_width()
            n_inf_now = len(self.infected_container)
            n_pop_now = len(self.all_container)
            n_rec_now = len(self.recovered_container)
            t = int((i / self.T) * stats_width)
            y_infect = int(
                stats_height - (n_inf_now / n_pop_now) * stats_height
            )
            y_dead = int(
                ((self.N - n_pop_now) / self.N) * stats_height
            )
            y_recovered = int((n_rec_now / n_pop_now) * stats_height)
            stats_graph = pygame.PixelArray(stats)
            stats_graph[t, y_infect:] = pygame.Color(*GREEN)
            stats_graph[t, :y_dead] = pygame.Color(*HORRIBLE_YELLOW)
            stats_graph[
                t, y_dead : y_dead + y_recovered
            ] = pygame.Color(*PURPLE)

            # New infections?
            collision_group = pygame.sprite.groupcollide(
                self.susceptible_container,
                self.infected_container,
                True,
                False,
            )

            for guy in collision_group:
                new_guy = guy.respawn(GREEN)
                new_guy.vel *= -1
                new_guy.killswitch(
                    self.cycles_to_fate, self.mortality_rate
                )
                self.infected_container.add(new_guy)
                self.all_container.add(new_guy)

            # Any recoveries?
            recovered = []
            for guy in self.infected_container:
                if guy.recovered:
                    new_guy = guy.respawn(PURPLE)
                    self.recovered_container.add(new_guy)
                    self.all_container.add(new_guy)
                    recovered.append(guy)

            if len(recovered) > 0:
                self.infected_container.remove(*recovered)
                self.all_container.remove(*recovered)

            self.all_container.draw(screen)

            del stats_graph
            stats.unlock()
            screen.blit(stats, stats_pos)
            pygame.display.flip()

            clock.tick(30)

        pygame.quit()


if __name__ == "__main__":
    covid = Simulation(1000, 480)
    covid.n_susceptible = 100
    covid.n_quarantined = 0
    covid.n_infected = 5
    covid.cycles_to_fate = 200
    covid.mortality_rate = 0.2
    covid.start(randomize=True)
```