```python
import pygame
import math

pygame.init()

WIDTH, HEIGHT = 800, 800
WIN = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Planet Simulation")

WHITE = (255, 255, 255)
YELLOW = (255, 255, 0)
BLUE = (100, 149, 237)
RED = (188, 39, 50)
DARK_GREY = (80, 78, 81)

FONT = pygame.font.SysFont("comicsans", 16)


class Planet:
    AU = 149.6e6 * 1000
    G = 6.67428e-11
    SCALE = 250 / AU
    TIMESTEP = 3600 * 24

    def __init__(self, x, y, radius, color, mass):
        self.x = x
        self.y = y
        self.radius = radius
        self.color = color
        self.mass = mass

        self.orbit = []
        self.sun = False
        self.distance_to_sun = 0

        self.x_vel = 0
        self.y_vel = 0

    def draw(self, win):
        x = self.x * self.SCALE + WIDTH / 2
        y = self.y * self.SCALE + HEIGHT / 2

        if len(self.orbit) > 2:
            updated_points = []
            for point in self.orbit:
                x, y = point
                x = x * self.SCALE + WIDTH / 2
                y = y * self.SCALE + HEIGHT / 2
                updated_points.append((x, y))

            pygame.draw.lines(win, self.color, False, updated_points, 2)

        pygame.draw.circle(win, self.color, (x, y), self.radius)

        if not self.sun:
            distance_text = FONT.render(f"{round(self.distance_to_sun /
1000, 1)}km", 1, WHITE)
```

```
            win.blit(distance_text, (x - distance_text.get_width() / 2, y
- distance_text.get_height() / 2))

    def attraction(self, other):
        other_x, other_y = other.x, other.y
        distance_x = other_x - self.x
        distance_y = other_y - self.y
        distance = math.sqrt(distance_x ** 2 + distance_y ** 2)

        if other.sun:
            self.distance_to_sun = distance

        force = self.G * self.mass * other.mass / distance ** 2
        theta = math.atan2(distance_y, distance_x)
        force_x = math.cos(theta) * force
        force_y = math.sin(theta) * force
        return force_x, force_y

    def update_position(self, planets):
        total_fx = total_fy = 0
        for planet in planets:
            if self == planet:
                continue

            fx, fy = self.attraction(planet)
            total_fx += fx
            total_fy += fy

        self.x_vel += total_fx / self.mass * self.TIMESTEP
        self.y_vel += total_fy / self.mass * self.TIMESTEP

        self.x += self.x_vel * self.TIMESTEP
        self.y += self.y_vel * self.TIMESTEP
        self.orbit.append((self.x, self.y))


def main():
    run = True
    clock = pygame.time.Clock()

    sun = Planet(0, 0, 30, YELLOW, 1.98892 * 10 ** 30)
    sun.sun = True

    earth = Planet(-1 * Planet.AU, 0, 16, BLUE, 5.9742 * 10 ** 24)
    earth.y_vel = 29.783 * 1000

    mars = Planet(-1.524 * Planet.AU, 0, 12, RED, 6.39 * 10 ** 23)
    mars.y_vel = 24.077 * 1000

    mercury = Planet(0.387 * Planet.AU, 0, 8, DARK_GREY, 3.30 * 10 ** 23)
    mercury.y_vel = -47.4 * 1000

    venus = Planet(0.723 * Planet.AU, 0, 14, WHITE, 4.8685 * 10 ** 24)
    venus.y_vel = -35.02 * 1000

    planets = [sun, earth, mars, mercury, venus]

    while run:
        clock.tick(60)
        WIN.fill((0, 0, 0))

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False

        for planet in planets:
            planet.update_position(planets)
            planet.draw(WIN)

        pygame.display.update()

    pygame.quit()


main()
```