```
from vpython import  *
 import math
```

vpython: Used for 3D visualizations in Python. It allows us to create spheres, lights, and animate them.

math: Provides math functions like sin, cos, radians, and pi.

```
scene = canvas(title='Earth-Moon-Mars Simulation',
               width=1200, height=800,
               background=color.black,
               center=vector(0,0,0))
```

canvas: Creates the 3D window where the simulation is displayed.

title: Name of the window.

width & height: Size of the canvas.

background=color.black: Makes the background space-like.

center=vector(0,0,0): Sets the camera to focus on the origin (0,0,0).

```
sun = sphere(pos=vector(0,0,0), radius=2, color=color.yellow, emissive=True)
light = local_light(pos=sun.pos, color=color.yellow)
sun_glow = sphere(pos=sun.pos, radius=4, color=color.yellow, opacity=0.1)
```

sphere: Represents a 3D object.

emissive=True: Makes the Sun appear as if it emits light.

local_light: Creates a point light at the Sun's position to illuminate other objects.

sun_glow: Semi-transparent sphere to give a glowing effect around the Sun.

```
# ----------------------------
earth = sphere(pos=vector(10,0,0), radius=1, color=color.blue, make_trail=True, trail_color=color.cyan)
moon = sphere(pos=vector(12,0,0), radius=0.27, color=color.white, make_trail=True, trail_color=color.white, retain=100)
mars = sphere(pos=vector(16,0,0), radius=0.87, color=color.red, make_trail=True, trail_color=color.yellow)
```

pos: Initial position in 3D space.

radius: Size of the sphere.

color: Sphere color.

make_trail=True: Leaves a trail to visualize the orbit.

retain=100: For Moon, only keeps last 100 points in the trail to avoid clutter.

```
moon_orbit_radius = 2
moon_tilt = radians(5)   # Moon orbit tilt
mars_tilt = radians(1.85) # Mars orbit tilt
```

moon_orbit_radius: Distance from Earth to Moon.

moon_tilt & mars_tilt: Tilt of the orbit in radians (realistic inclination).

```
t = 0
dt = 0.5  # base time step (days)
```

t: Simulation time in days.

dt: Base time step for each frame.

```
earth_period = 365.0
moon_period = 27.3
mars_period = 687.0
```

*_period: Real orbital periods in days.

```
# Angular speeds (radians/day)
omega_earth = 2*pi / earth_period
omega_moon = 2*pi / moon_period
omega_mars = 2*pi / mars_period
```

omega_*: Angular speed (radians per day) using formula:

$$\omega = \frac{2\pi}{T}$$

```
earth_angle = 0
moon_angle = 0
mars_angle = 0
```

*_angle: Current angle of the object in its orbit.

```
speed_multiplier = 0.1

def update_speed(s):
    global speed_multiplier
    speed_multiplier = s.value

wtext(text='Simulation speed multiplier: ')
speed_slider = slider(min=0.1, max=50, value=1, length=300, bind=update_speed, right=15)
speed_text = wtext(text=f' {speed_multiplier:.1f}x')

def update_slider_text():
    speed_text.text = f' {speed_multiplier:.1f}x'
```

Allow to change the simulation speed interactively.

bind=update_speed: Calls update_speed function whenever slider is moved.

Updates the displayed multiplier on screen.

```
date_text = wtext(text='\nDay: 0\n')

def update_date_text():
    year = int(t // 365) + 1
    day_of_year = int(t % 365) + 1
    date_text.text = f'\nYear: {year}, Day: {day_of_year}'
```

Shows the simulated day and year based on t.

Helps track time in the simulation.

```
# ----------------------------
while True:
    rate(200)

    # Increment angles using speed multiplier
    earth_angle += omega_earth * dt * speed_multiplier
    moon_angle += omega_moon * dt * speed_multiplier
    mars_angle += omega_mars * dt * speed_multiplier
    t += dt * speed_multiplier

    # Update Earth position
    earth.pos = vector(10*cos(earth_angle), 10*sin(earth_angle), 0)

    # Update Moon position (tilted orbit around Earth)
    moon.pos = earth.pos + vector(
        moon_orbit_radius * cos(moon_angle),
        moon_orbit_radius * sin(moon_angle) * cos(moon_tilt),
        moon_orbit_radius * sin(moon_angle) * sin(moon_tilt)
    )

    # Update Mars position (tilted orbit around Sun)
    mars.pos = vector(
        16*cos(mars_angle),
        16*sin(mars_angle)*cos(mars_tilt),
        16*sin(mars_angle)*sin(mars_tilt)
    )

    # Update slider text and date display
    update_slider_text()
    update_date_text()
```

rate(200): Limits simulation to 200 updates per second.

Updates angles and time according to orbital speed and multiplier.

Earth moves in a circular orbit around the Sun.

Moon orbits Earth with a small tilt for realism.

Mars orbits Sun with a slight tilt.

Updates slider display and simulation date.

x: position along the orbit in the plane.
y: projected into the orbital plane, adjusted by tilt.
z: height above/below the plane, depending on tilt.

x = R * cos(theta)
y = R * sin(theta) * cos(tilt)
z = R * sin(theta) * sin(tilt)

Summary:

This code simulates a mini solar system with the Sun, Earth, Moon, and Mars.
It includes realistic orbital periods, orbit tilts, trails,and a speed control slider.
The animation shows planets revolving around the Sun and the Moon revolving around Earth in 3D space.