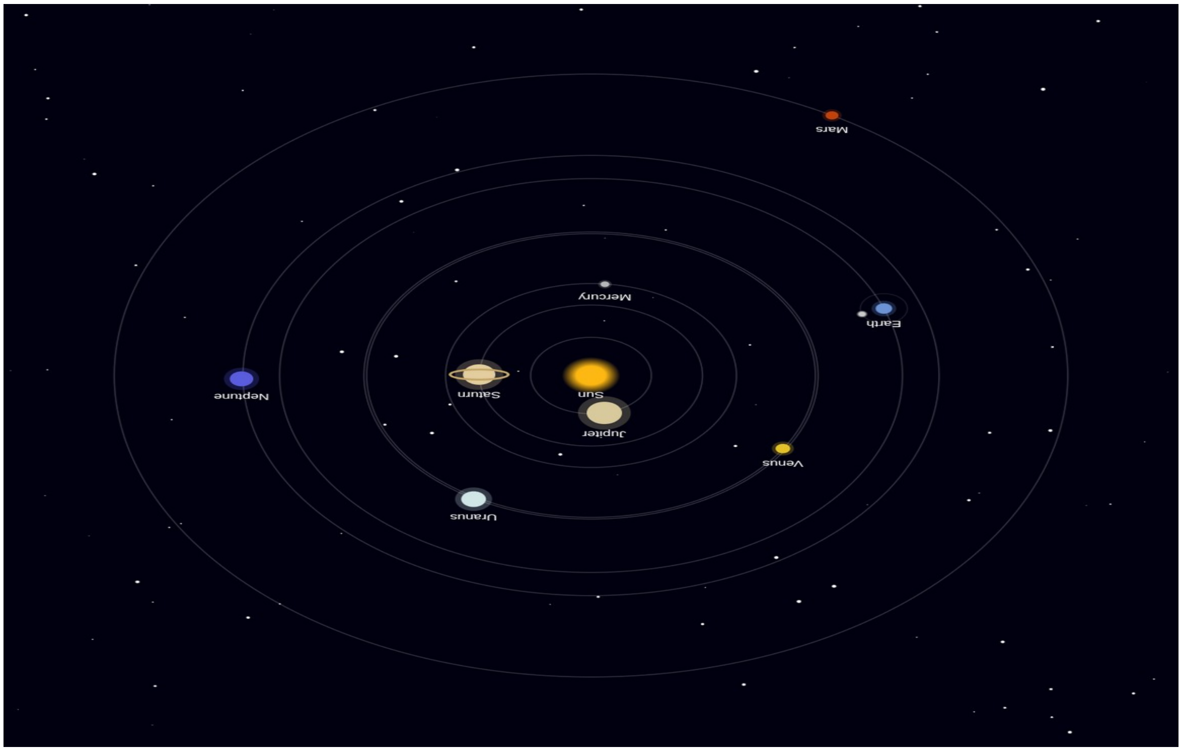

🪐 Document: Solar System Position Calculation in solarsystem.js

The solarsystem.js file uses **Keplerian orbital elements** and classical celestial mechanics, adapted from sources like *Jean Meeus's Astronomical Algorithms*, to compute the heliocentric (Sun-centered) positions of the planets for a given moment in time. The code combines fundamental astronomical constants, time-dependent orbital equations, and an iterative solver for Kepler's Equation.

---

1. 🕰️ Time Calculation

The first critical step is to determine the Julian Ephemeris Day offset, D, which is the number of days (including fractional days) from a standard epoch (reference time) to the time of interest.

- **Epoch Definition:** The calculations in this file are relative to an epoch near J2000.0 (January 1, 2000, at 12:00 TT). The specific constant -730530 in the getD() function defines the epoch as **January 0.5, 2000, 12:00 UT**.
- **getD() Function:**

- It hardcodes the time to **October 12, 2025, 23:21:00 UTC** (based on new Date('2025-10-12T23:21:00Z')).
- It calculates the day count D since the epoch using a standard formula for converting calendar dates to Julian Day number equivalents, then subtracts the reference Julian Day number, 730530, and adds the fraction of the day $ut/24$.
- **Result:** The variable D is the number of **Julian Ephemeris Days** (JED) since the epoch, which is used as the input for all subsequent orbital element calculations.

---

2. 🔢 Orbital Elements and Time Dependence

The planetsData array holds the data for each body. For the planets, the elements property is a function that takes D as an argument and returns the six **classical Keplerian orbital elements**:

- **N** (Ω): **Longitude of the Ascending Node** (degrees)
- **i**: **Inclination** (degrees)
- **w** (ω): **Argument of Perihelion** (degrees)
- **a**: **Semi-major Axis** (Astronomical Units, AU)
- **e**: **Eccentricity** (dimensionless)
- **M**: **Mean Anomaly** (degrees)

The first five elements define the **shape and orientation** of the elliptical orbit in space, while the Mean Anomaly (M) defines the **position** of the planet *along* that orbit at time D.

Most of these elements are expressed as **linear functions of D**: $Element = BaseValue + Rate \times D$. This accounts for the slow, long-term changes (called **secular perturbations**) in the orbits over time.

---

3. 🎯 Kepler's Equation Solver

To find the true position of a planet in its orbit from the Mean Anomaly (M), two intermediate steps are required:

A. Solving Kepler's Equation

The Mean Anomaly (M) is the position the planet *would* have if it moved in a perfect circle at a constant speed. To get the position in an ellipse, the **Eccentric Anomaly** (E) must be found by solving **Kepler's Equation**:

$$M = E - e \cdot \sin(E)$$

The solveKepler(M, e) function performs this calculation:

1. It uses an approximate starting value for E based on the first two terms of its series expansion.
2. It then uses the **Newton-Raphson method** (an iterative approximation technique) for $e > 0.05$ (Mercury, Mars, Jupiter, Saturn) to refine E until the change (Δ) is less than 1×10^{-6} radians or 100 iterations are performed.
3. **Result:** The function returns E in degrees.

B. Calculating the True Anomaly and Radial Distance

Using E and the eccentricity e, the coordinates within the plane of the orbit are found:

- $x_v = a \cdot (\cos(E) - e)$
- $y_v = a \cdot \sqrt{1 - e^2} \cdot \sin(E)$

From these:

- **Radial Distance (r):** The distance from the Sun to the planet: $r = \sqrt{x_v^2 + y_v^2}$
- **True Anomaly (v):** The actual angular position of the planet in its orbit plane, measured from perihelion (closest approach to the Sun): $v = \text{atan2}(y_v, x_v)$

---

4. 🌍 Heliocentric Cartesian Coordinates

The getHelioPos(d, pd) function converts the planet's position from its orbital plane to the **ecliptic plane** (the plane of Earth's orbit, $i=0$ for Earth) using three Euler rotations. This yields the final 3D position relative to the Sun (at the origin, 0,0,0) in Astronomical Units (AU):

- **Rotation 1:** By $(\omega + v)$ to place the planet relative to the Node.

- **Rotation 2:** By $-i$ to account for the orbit's inclination.
- **Rotation 3:** By $-\Omega$ to align the Node with the Ecliptic x-axis.

The resulting cartesian coordinates are:

- $x_h = r \cdot (\cos(\Omega) \cos(v+\omega) - \sin(\Omega) \sin(v+\omega) \cos(i)) $
- $y_h = r \cdot (\sin(\Omega) \cos(v+\omega) + \cos(\Omega) \sin(v+\omega) \cos(i))$
- $z_h = r \cdot \sin(v+\omega) \sin(i)$

The function also calculates the **Heliocentric Longitude (lon) and Latitude (lat) in the Ecliptic Frame** from these coordinates.

---

5. 📏 Perturbations (Gravitational Adjustments)

A key detail for achieving higher accuracy is the inclusion of short-term gravitational effects between the giant planets (**Jupiter, Saturn, Uranus**). These are called **perturbations**.

The getHelioPos function includes a switch statement that applies corrections (Δlon and Δlat) to the planet's longitude and latitude. These corrections are based on a Fourier-series-like summation involving the Mean Anomalies of the mutually perturbing planets (e.g., Jupiter's position is corrected based on the mean longitudes of Saturn and Jupiter).

- **Final Position:** The final x_h, y_h, z_h coordinates are then recalculated using the perturbed lon and lat values.

---

6. 🖼️ Visualization and Scaling

The rest of the code is responsible for rendering the calculated positions on a 2D canvas:

- **Scaling:** The positions (in AU) are converted to pixel coordinates for display. Since the orbits of the inner (rocky) planets are vastly different from the outer (gas giant) planets, a dual scaling approach is used:
    - Inner planets (Mercury to Mars) are scaled to fit a view up to 1.6 AU (innerMaxR).

    - Outer planets are scaled by a smaller factor (multiplied by a ratio of $1.6/30 \cdot 0.7$) to keep them visible within the same screen area, sacrificing accurate distance representation for visual completeness.
- **Drawing:** The x_h and y_h coordinates are plotted on the canvas using the calculated scale. The canvas is rotated -180° to likely orient the display with North up and the Ecliptic x-axis to the right.
- **Moon:** The Moon's position is not calculated using Keplerian elements but is simply a circle rotating around the Earth's position at a fixed distance and speed (moonAngle += moon.speed), which is a simplified, non-real-time approximation.