3D Earth and Celestial Bodies

MATLAB Implementation

Tamas Kis | kis@stanford.edu

TAMAS KIS
https://github.com/tamaskis

LICENSE

Contents

planet3D

Creates high-resolution renderings of the Earth and the major celestial bodies in our solar system for space mechanics applications.

Syntax

```
planet3D(planet,position,gmst,reference_plane,units,transparency)
```

NOTE: All parameters except for `planet` are optional. If you "skip over" parameters, you need to use empty bracket (i.e. "`[]`") as placeholders, otherwise you can emit parameters altogether. For example, if you don't want to specify `position`, but *do* want to specify `units`, then you would use the syntax `planet3D(planet,[],[],[],units)`. Alternatively, if we wanted to specify just the `position`, we could use the syntax `planet3D(planet,position)`.

NOTE: Use the `background` function (see Section) to set the plot background. When using `background` to set the plot background, the function call on `background` must occur *before* the function call on `planet3D`, otherwise the background will be plotted *over* the celestial body.

Description

`planet3D(planet,position,gmst,reference_plane,units,transparency)` draws a celestial body.

`planet`	`'Sun'`, `'Moon'`, `'Mercury'`, `'Venus'`, `'Earth'`, `'Earth Cloudy'`, `'Earth Night'`, `'Earth Night Cloudy'`, `'Mars'`, `'Jupiter'`, `'Saturn'`, `'Uranus'`, `'Neptune'`, or `'Pluto'`.
`position` *(optional)*	Specifies the position of the celestial body. If `position` is not specified, the function defaults to $(0, 0, 0)$. NOTE: If you are also specifying `units`, make sure you input `position` in the correct units (i.e. in the units you intend to use).
`gmst` *(optional)*	Specifies the Greenwich mean sidereal time (the angle from the direction of the vernal equinox to 0 degrees longitude, measured in degrees).
`reference_plane` *(optional)*	Specifies which reference plane the celestial body is drawn with respect to. If specified as `'equatorial'`, the reference plane is taken to be the equatorial plane of the celestial body. If specified as `'ecliptic'`, the celestial body will be tilted by the obliquity (i.e. the angle between the ecliptic plane and the equatorial plane).
`units` *(optional)*	Specifies the units the celestial body should be drawn in. Units available are `'km'`, `'AU'`, `'m'`, `'ft'`, `'mi'`, and `'nmi'`.
`transparency` *(optional)*	Specifies how transparent the celestial body is (0 for 100% transparency, 1 for 100% solid).

Links

MATLAB® Central's File Exchange:
https://www.mathworks.com/matlabcentral/fileexchange/86483-3d-earth-and-celestial-bodies-planet3d

GitHub®:
https://github.com/tamaskis/planet3D-MATLAB

background

Sets the plot background for drawing celestial bodies in 3D.

Syntax

```
background(spec)
```

NOTE: The function call on `background` must occur before the function call on `planet3D`.

Description

`background(spec)` sets the plot background for drawing celestial bodies in 3D. `spec` refers to the specified background, and can be set to `'Black'`, `'Stars'`, or `'Milky Way'`.

Links

MATLAB® Central's File Exchange:
https://www.mathworks.com/matlabcentral/fileexchange/86483-3d-earth-and-celestial-bodies-planet3d

GitHub®:
https://github.com/tamaskis/planet3D-MATLAB

Example Plots

Celestial Bodies on Milky Way Background

Earth (With Clouds)

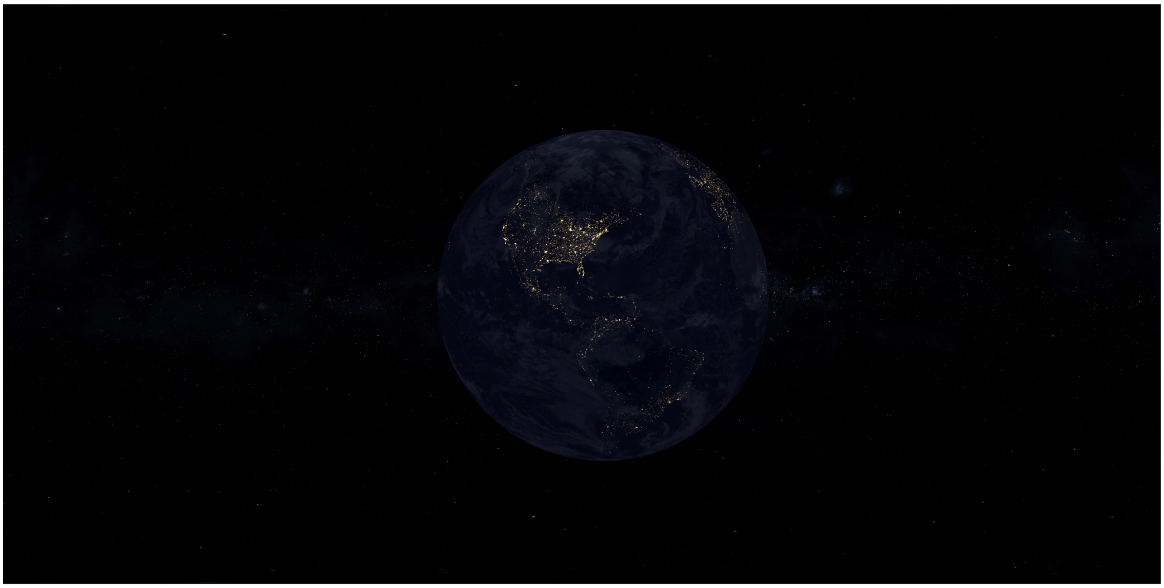
```
figure;
background('Milky Way');
planet3D('Earth Cloudy');
light('Position',[1,-1,0]);
```

Earth (No Clouds)

```
figure;
background('Milky Way');
planet3D('Earth')
light('Position',[1,-1,0]);
```

Earth (Night, With Clouds)

```
figure;
background('Milky Way');
planet3D('Earth Night Cloudy')
light('Position',[1,-1,0]);
```



Earth (Night, No Clouds)

```
figure;
background('Milky Way');
planet3D('Earth Night')
```

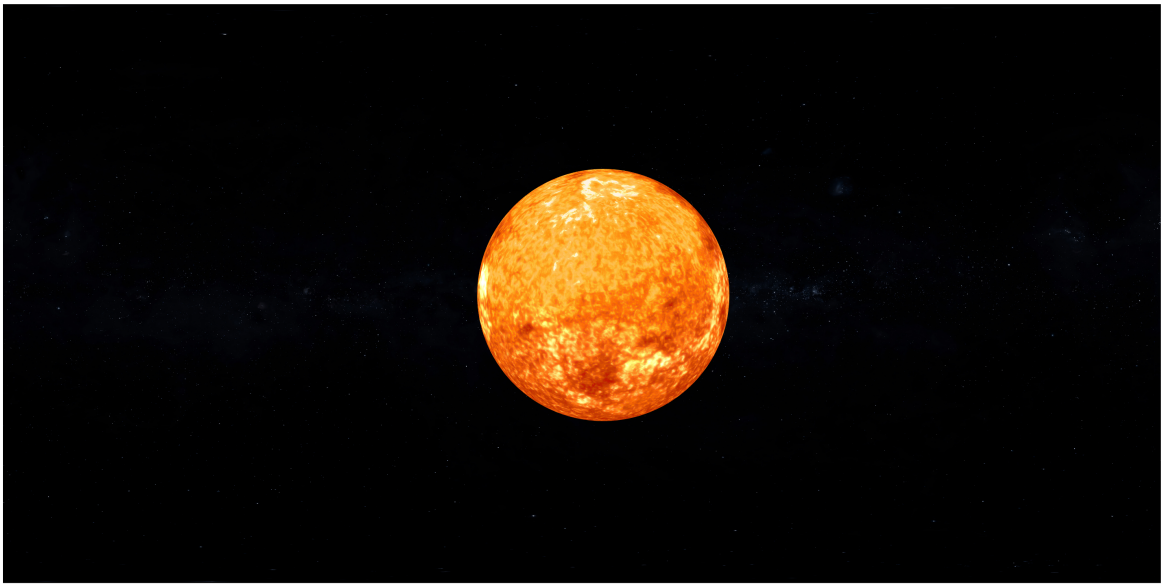
```
light('Position',[1,-1,0]);
```

Moon

```
figure;
background('Milky Way');
planet3D('Moon')
light('Position',[1,-1,0]);
```

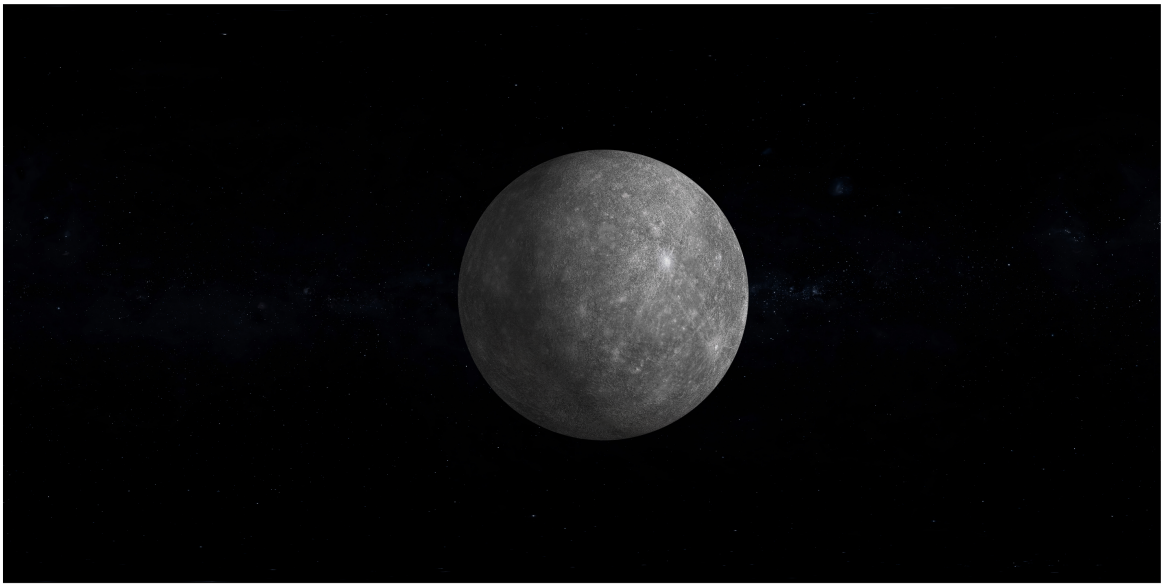
Sun

```
figure;
background('Milky Way');
planet3D('Sun')
light('Position',[1,-1,0]);
```



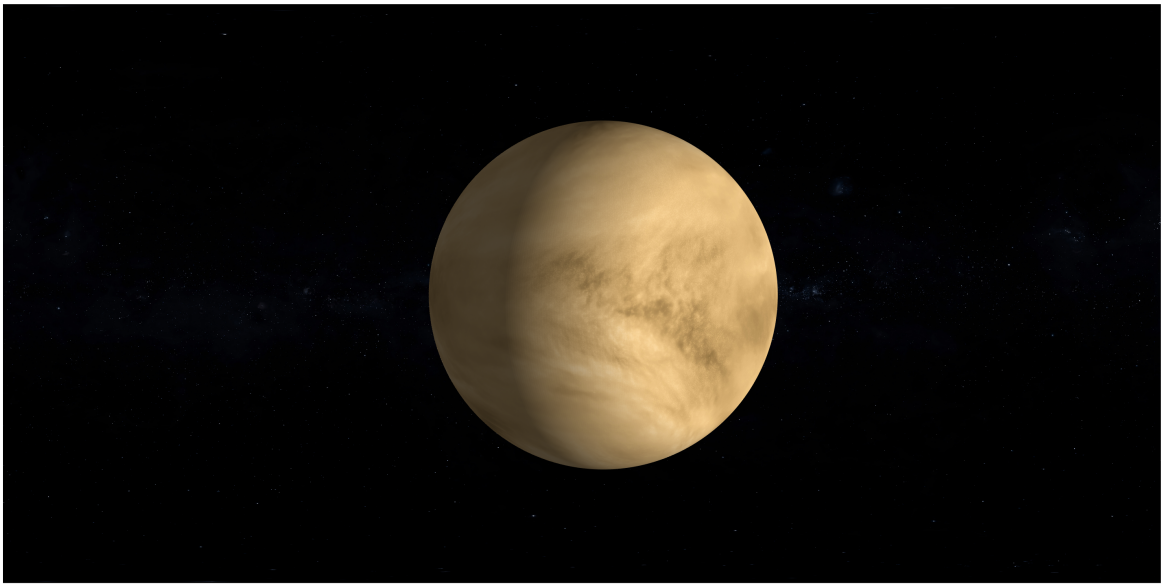
Mercury

```
figure;
background('Milky Way');
planet3D('Mercury')
light('Position',[1,-1,0]);
```



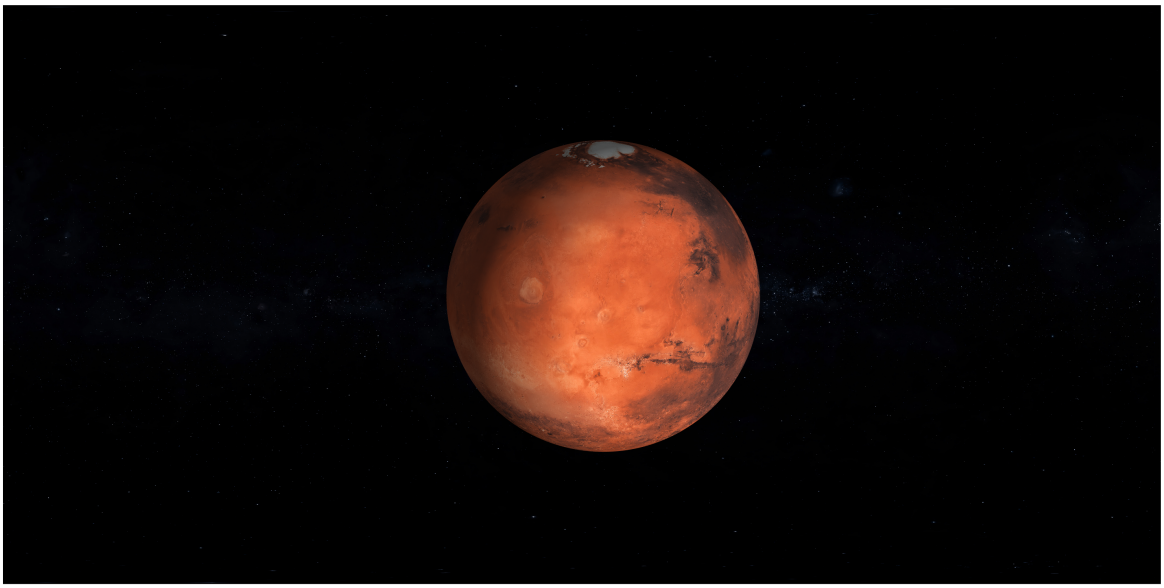
Venus

```
figure;
background('Milky Way');
planet3D('Venus')
light('Position',[1,-1,0]);
```



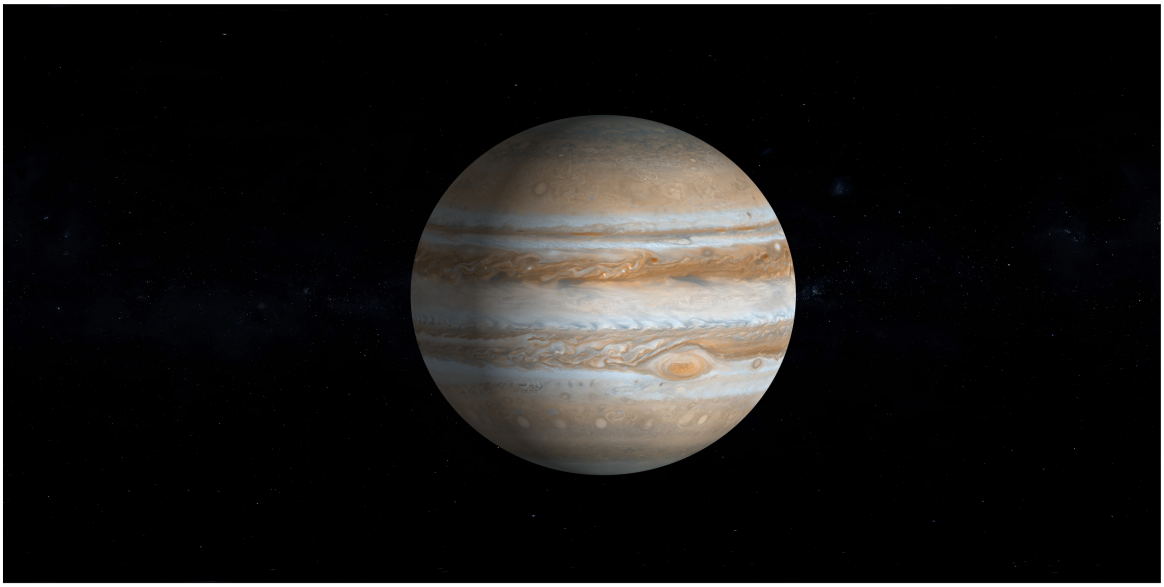
Mars

```
figure;
background('Milky Way');
planet3D('Mars')
light('Position',[1,-1,0]);
```



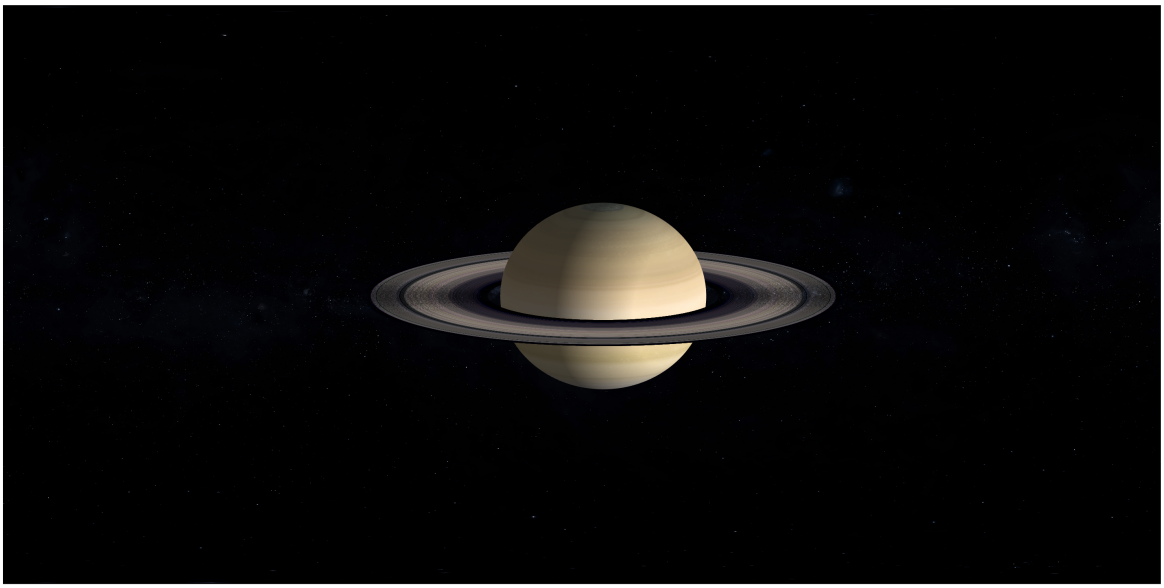
Jupiter

```
figure;
background('Milky Way');
planet3D('Jupiter')
light('Position',[1,-1,0]);
```



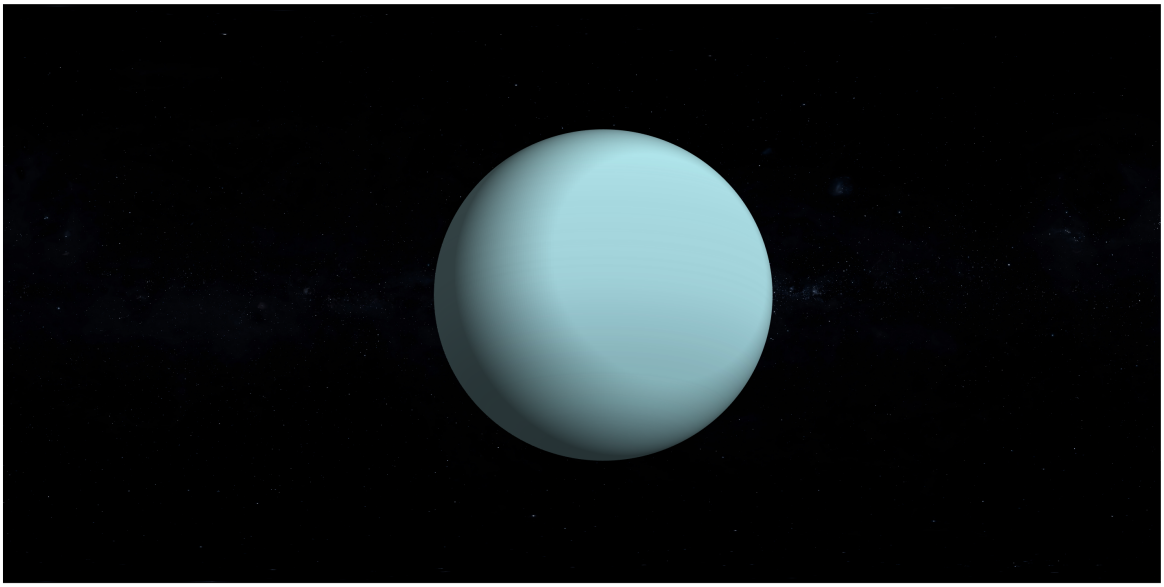
Saturn

```
figure;
background('Milky Way');
planet3D('Saturn')
light('Position',[1,-1,0]);
```



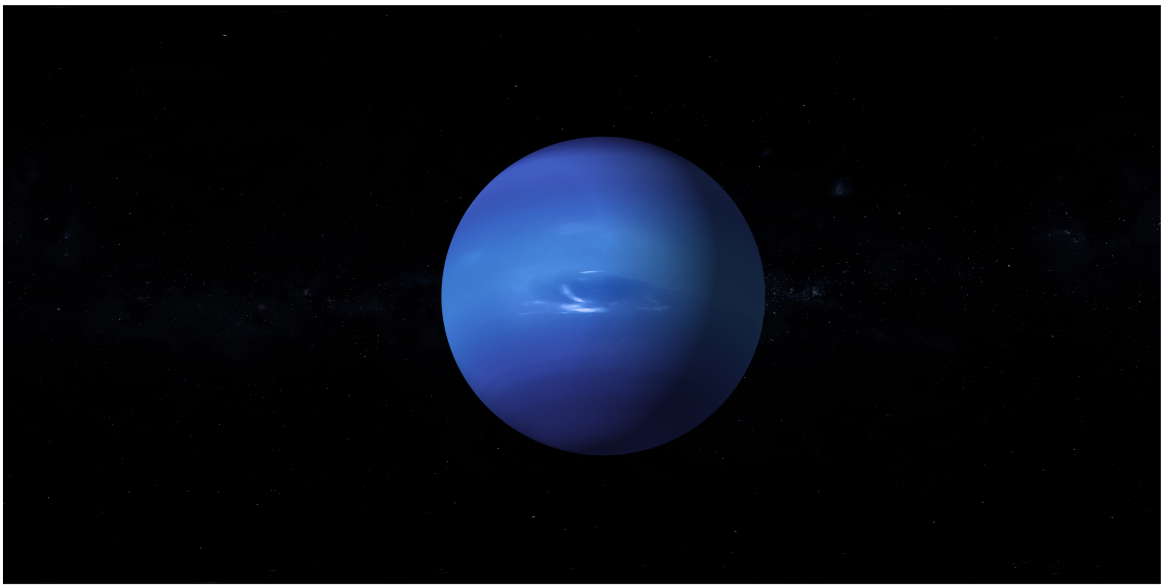
Uranus

```
figure;
background('Milky Way');
planet3D('Uranus')
light('Position',[1,-1,0]);
```



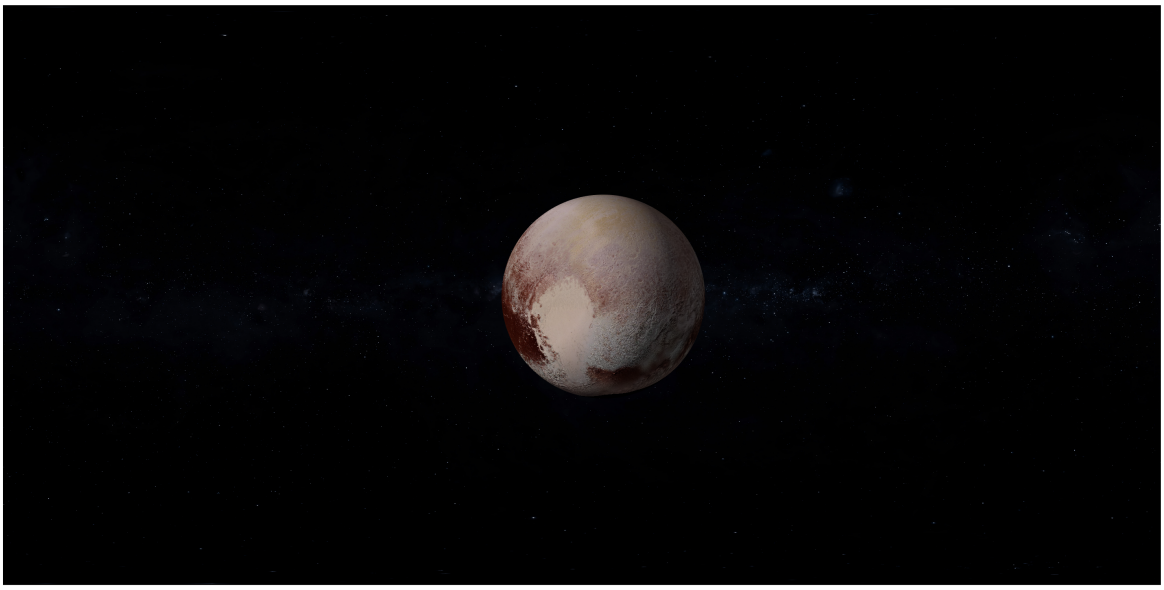
Neptune

```
figure;
background('Milky Way');
planet3D('Neptune')
light('Position',[1,-1,0]);
```



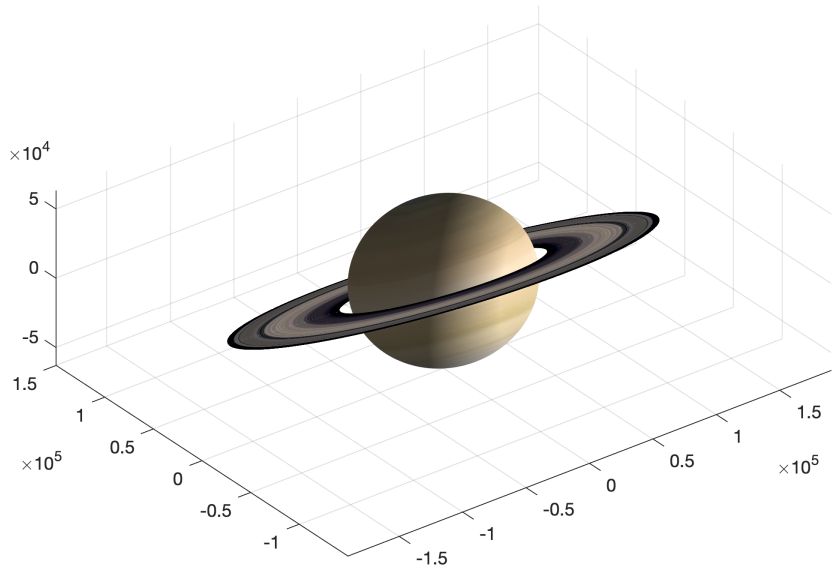
Pluto

```
figure;
background('Milky Way');
planet3D('Pluto')
light('Position',[1,-1,0]);
```



Saturn in Ecliptic Plane on White Background with Grid Lines

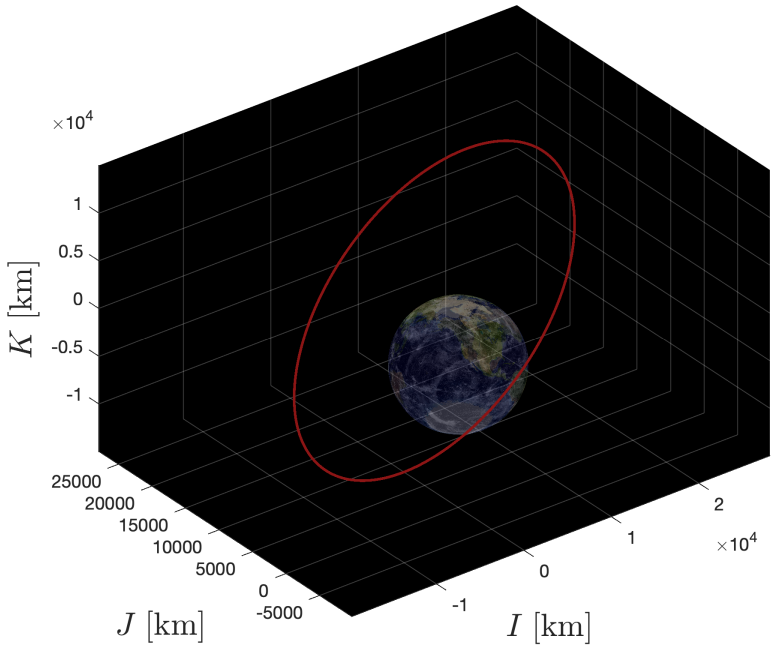
```
figure;
planet3D('Saturn',[],[],'ecliptic')
light('Position',[1,-1,0]);
grid on;
```



Elliptical Trajectory Around Transparent Earth on Black Background with Grid Lines

```
% sets orbit parameters
p = 15000; % semi-latus rectum [km]
e = 0.5; % eccentricity
i = 45; % inclination [deg]
Om = 245; % right ascension of the ascending node [deg]
w = 0; % argument of periapsis [deg]

% orbit in perifocal plane
nu = 0:0.01:(2*pi); % true anomaly [rad]
r_P = p*cos(nu)./(1+e*cos(nu));
r_Q = p*sin(nu)./(1+e*cos(nu));
r_W = zeros(size(nu));

% rotation matrix from perifocal frame to Earth-centererd inertial frame
sO = sind(Om);
cO = cosd(Om);
si = sind(i);
ci = cosd(i);
sw = sind(w);
cw = cosd(w);
R_PQW_to_IJK = [cO*cw-sO*ci*sw   -cO*sw-sO*ci*cw   sO*si;
                sO*cw+cO*ci*sw   -sO*sw+cO*ci*cw   -cO*si;
                si*sw            si*cw             ci];

% rotates coordinates of orbit to ECI frame
rI = zeros(size(nu));
rJ = zeros(size(nu));
rK = zeros(size(nu));
for j = 1:length(nu)
    new_coordinates = R_PQW_to_IJK*[r_P(j);r_Q(j);r_W(j)];
    rI(j) = new_coordinates(1);
    rJ(j) = new_coordinates(2);
```

```
    rK(j) = new_coordinates(3);
end

% plot
figure;
background('Black');
hold on;
planet3D('Earth Cloudy',[],[],[],[],0.25);
plot3(rI,rJ,rK,'color',[140,21,21]/255,'linewidth',1.5);
hold off;
grid on;
ax = gca;
ax.GridColor = [1,1,1];
ax.GridAlpha = 0.25;
xlabel("$I\;[\mathrm{km}]$",'interpreter','latex','fontsize',18);
ylabel("$J\;[\mathrm{km}]$",'interpreter','latex','fontsize',18);
zlabel("$K\;[\mathrm{km}]$",'interpreter','latex','fontsize',18);
```



Data and Constants

Astronomical Data

Planet/Body	Equatorial Radius		Flattening		Obliquity	
	Value [km]	Source	Value	Source	Value [°]	Source
Sun	696000	[13]*	0.000 009	[11]	0	-
Moon	1738.0	[13]**	0.0012	[7]	6.68	[13]**
Mercury	2439.0	[13]**	0.0000	[6]	0.0	[13]**
Venus	6052.0	[13]**	0.000	[14]	177.3	[13]**
Earth	6378.1363	[13]**	0.003 352 813 1	[13]**	23.45	[13]**
Mars	3397.2	[13]**	0.006 476 30	[13]**	25.19	[13]**
Jupiter	71492.0	[13]***	0.064 874 4	[13]***	3.12	[13]***
Saturn	60268.0	[13]***	0.097 962 4	[13]***	26.73	[13]***
Uranus	25559.0	[13]***	0.022 927 3	[13]***	97.86	[13]***
Neptune	24764.0	[13]***	0.0171	[13]***	29.56	[13]***
Pluto	1151.0	[13]***	0.0	[13]***	118.0	[13]***

**Table D-5, p. 1043*
***Table D-3, p. 1041*
****Table D-4, p. 1042*

Semi-Minor Axes

For MATLAB's `ellipsoid` function, we need the semi-minor axis, b, which can be calculated as

$$b = a(1 - f)$$

where a is the semi-major axis (assumed to be the equatorial radius) and f is the flattening [3, p. 7-4 (p. 73 in PDF)].

Saturn's Rings

Saturn's rings range from $7000\ \text{km}$ to $80000\ \text{km}$ from the surface of the planet [9].

Unit Conversions

Kilometers to Astronomical Units [13]:

$$1\ \text{AU} = 149597870\ \text{km} \quad \rightarrow \quad \boxed{1\ \text{km} = \frac{1}{149597870}\ \text{AU}}$$

Kilometers to Meters:

$$\boxed{1\ \text{km} = 1000\ \text{m}}$$

Kilometers to Feet:

$$1\text{ km} = \left(\frac{1000\text{ m}}{1\text{ km}}\right)\left(\frac{100\text{ cm}}{1\text{ m}}\right)\left(\frac{1\text{ in}}{2.54\text{ cm}}\right)\left(\frac{1\text{ ft}}{12\text{ in}}\right) \quad \rightarrow \quad \boxed{1\text{ km} = \frac{100000}{30.48}\text{ ft}}$$

Kilometers to Miles:

$$1\text{ km} = \left(\frac{100000/30.48\text{ ft}}{\text{km}}\right)\left(\frac{1\text{ mi}}{5280\text{ ft}}\right) \quad \rightarrow \quad \boxed{1\text{ km} = \frac{100000}{160934.4}\text{ mi}}$$

Kilometers to Nautical Miles:

$$1\text{ nmi} = 1852\text{ m} = 1.852\text{ km} \quad \rightarrow \quad \boxed{1\text{ km} = \frac{1}{1.852}\text{ nmi}}$$

Image Sources

Image	File Name	Source	Copyright/License
Sun	`Sun.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Moon	`Moon.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Mercury	`Mercury.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Venus	`Venus.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Earth (Day)	`Earth.png`	[12]	none [5, 12]
Earth (Night)	`Earth Night.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Clouds	`Clouds.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Mars	`Mars.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Jupiter	`Jupiter.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Saturn	`Saturn.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Saturn Rings	`Saturn Rings.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Uranus	`Uranus.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Neptune	`Neptune.png`	[10]	CC Attribution 4.0 International (CC BY 4.0) [1, 10]
Pluto	`Pluto.png`	[8]	none [8]
Milky Way	`Milky Way.png`	[10]	CC Attribution 4.0 International (CC BY 4.0)
Stars	`Stars.png`	[10]	CC Attribution 4.0 International (CC BY 4.0)

References for Code

3D Earth Example (`earth_example.m`) [4]:

- Use of `ellipsoid` function to render the Earth.

Earth-sized Sphere with Topography (`earth_sphere`) [2]:

- Handling of unit conversions.

References

[1] *Attribution 4.0 International (CC BY 4.0)*. creative commons. https://creativecommons.org/licenses/by/4.0/. (accessed: January 27, 2021).

[2] Will Campbell. *Earth-sized Sphere with Topography*. MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/27123-earth-sized-sphere-with-topography. (accessed: January 22, 2021).

[3] *Department of Defense World Geodetic System 1984*. Tech. rep. NIMA TR8350.2. https://apps.dtic.mil/sti/pdfs/AD1000581.pdf. National Imagery and Mapping Agency, 2004.

[4] Ryan Gray. *3D Earth Example*. MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/13823-3d-earth-example. (accessed: January 22, 2021).

[5] *Image Use Policy*. NASA visible earth. https://visibleearth.nasa.gov/image-use-policy. (accessed: January 23, 2021).

[6] *Mercury Fact Sheet*. NASA. https://nssdc.gsfc.nasa.gov/planetary/factsheet/mercuryfact.html. (accessed: January 22, 2021).

[7] *Moon*. Wikipedia. https://en.wikipedia.org/wiki/Moon. (accessed: January 22, 2021).

[8] *Pluto Color Map*. NASA Jet Propulsion Laboratory. https://www.jpl.nasa.gov/images/pluto-color-map/. (accessed: January 23, 2021).

[9] *Rings of Saturn*. Wikipedia. https://en.wikipedia.org/wiki/Rings_of_Saturn. (accessed: January 22, 2021).

[10] *Solar Textures*. Solar System Scope. https://www.solarsystemscope.com/textures/. (accessed: January 22, 2021).

[11] *Sun*. Wikipedia. https://en.wikipedia.org/wiki/Sun. (accessed: January 22, 2021).

[12] *The Blue Marble: Land Surface, Ocean Color and Sea Ice*. NASA visible earth. https://visibleearth.nasa.gov/images/57730/the-blue-marble-land-surface-ocean-color-and-sea-ice/57731l. (accessed: January 22, 2021).

[13] David A. Vallado. *Fundamentals of Astrodynamics and Applications*. 4th. Hawthorne, CA: Microcosm Press, 2013.

[14] *Venus Fact Sheet*. NASA. https://nssdc.gsfc.nasa.gov/planetary/factsheet/venusfact.html. (accessed: January 22, 2021).