Source Code

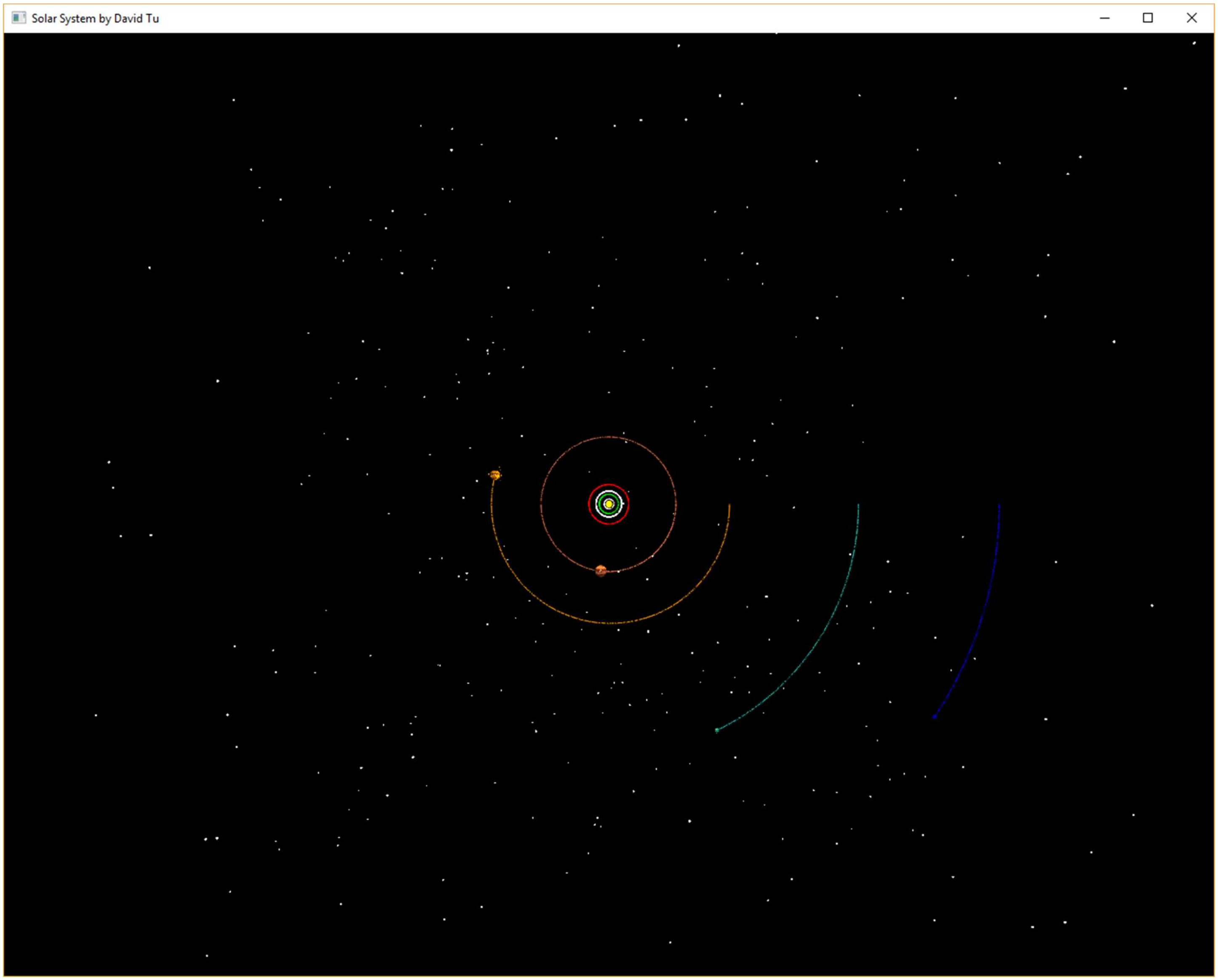
```
'''
File: Solar System.py

Name: David Tu

Program Description: Solar System Simulation: This program demonstrates our Solar System
It also includes rings for Jupiter, Saturn, Uranus and Neptune and shows the orbit of our Earth and Moon
To use this program, simply run this program with VIDLE, the IDE for Visual Python
Please note that the planets are larger than they should be but can be reverted back to their actual sizes by setting
'EarthMoonView = True'
When to scale, you should also be able to view our Moon orbiting around the Earth by zooming in
'''
from visual import *
from random import *

EarthMoonView = False    # Use this to switch between viewing the entire solar system and viewing the Earth-Moon orbits
                         # if EarthMoonView is switched on (i.e. EarthMoonView = True), zoom in to see the Earth and Moon
                         # to observe their orbits

scene = display(title = "Solar System by David Tu", width = 1280, height = 1024)
G = 6.674e-11 # Gravitational Constant

if EarthMoonView:
    planetScale = 1
    sunScale = 1
else:
    planetScale = 1000 # Adjust to scale up planet size in order to make it visible
    sunScale = 60 # Adjust to scale up sun size in order to make it visible

def createAxes(length): # Make axes visible (of world)
    directions = [vector(length, 0, 0), vector(0, length, 0), vector(0, 0, length)]
    texts = ["X", "Y", "Z"]
    position = vector(0, 0, 0)
    for i in range (3): # For X, Y, and Z:
       curve(pos = [position, position + directions[i]], color = color.white) # Create a line with a start and end point
       label(pos = position + directions[i], text = texts[i], color = color.white, opacity = 0, box = False)

#createAxes(1e12) # Used for debugging

'''
Refer to the following for the planetary data used:
    https://nssdc.gsfc.nasa.gov/planetary/factsheet/
    https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html
'''
# Sun setup (in meters)
LightSource = local_light(pos = (0, 0, 0), color = color.yellow)
Sun = sphere(pos = (0, 0, 0), radius = 6.957e8 * sunScale, color = color.yellow, material = materials.emissive)

# Inner planets setup (in meters)
Mercury = sphere(pos = (5.8e10, 0, 0), radius = 2.4395e6 * planetScale, color = color.white, material = materials.marble,
make_trail = False)
Venus = sphere(pos = (1.1e11, 0, 0), radius = 6.052e6 * planetScale, color = color.green, material = materials.marble,
make_trail = False)
Earth = sphere(pos = (1.5e11, 0, 0), radius = 6.378e6 * planetScale, color = color.cyan, material = materials.earth,
make_trail = False)
Moon = sphere(pos = Earth.pos + (0.384e9, 0, 0), radius = 1.740e6 * planetScale, color = color.white,
material = materials.marble, make_trail = False)   # Moon's position is the distance of the Earth from the Sun
                                        # plus distance of the Moon from the Earth
Mars = sphere(pos = (2.3e11, 0, 0), radius = 3.396e6 * planetScale, color = color.red, material = materials.marble,
make_trail = False)

# Outer planets setup (in meters)
Jupiter = sphere(pos = (7.8e11, 0, 0), radius = 7.1492e7 * planetScale, color = vector(norm(vector(255, 127, 80))),
material = materials.marble, make_trail = False)
JupiterRing = ring(pos = Jupiter.pos, color = Jupiter.color, axis = (0, 0, 1), radius = Jupiter.radius * 2,
thickness = 1e8)
Saturn = sphere(pos = (1.4e12, 0, 0), radius = 6.0268e7 * planetScale, color = color.orange, material = materials.marble,
make_trail = False)
SaturnRing = ring(pos = Saturn.pos, color = Saturn.color, axis = (0, 0, 1), radius = Saturn.radius * 2, thickness = 1e9)
Uranus = sphere(pos = (2.9e12, 0, 0), radius = 2.5559e7 * planetScale, color = vector(norm(vector(32, 178, 170))),
material = materials.marble, make_trail = False)
UranusRing = ring(pos = Uranus.pos, color = Uranus.color, axis = (1,0,0), radius = Uranus.radius * 2, thickness = 1e9)
```

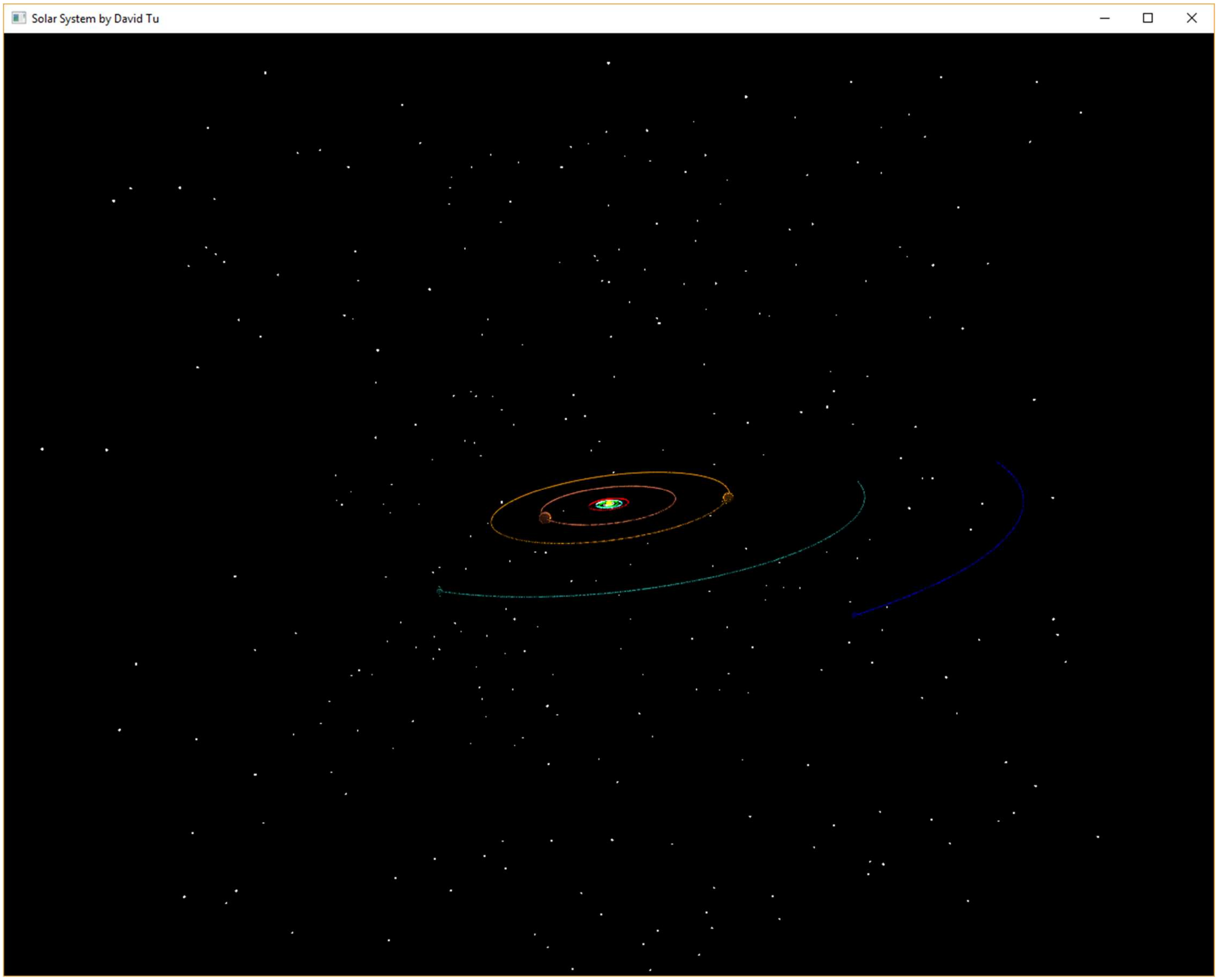
```python
Neptune = sphere(pos = (4.5e12, 0, 0), radius = 2.4764e7 * planetScale, color = color.blue, material = materials.marble,
make_trail = False)
NeptuneRing = ring(pos = Neptune.pos, color = Neptune.color, axis = (0, 0, 1), radius = Neptune.radius * 2,
thickness = 1e8)

# Generate randomly distanced stars as far as Neptune
for star in range (300):
    sphere(pos = (randint(-Neptune.pos.x, Neptune.pos.x), randint(-Neptune.pos.x, Neptune.pos.x), randint(-Neptune.pos.x,
Neptune.pos.x)), radius = 1e10, color = color.white, material = materials.emissive)

# Orbital Inclination setup
# Negatives are used to make the rotation go counter-clockwise in the x-z plane. Y is the axis of rotation
Mercury.rotate(angle = -7 * (pi/180), origin = (0, 0, 0), axis = (0, 1, 0)) # The orbital inclination of Mercury is 7
                                                                          # degrees but the angle is converted to radians
Venus.rotate(angle = -3.4 * (pi/180), origin = (0, 0, 0), axis = (0, 1, 0))
Mars.rotate(angle = -1.9 * (pi/180), origin = (0, 0, 0), axis = (0, 1, 0))
Jupiter.rotate(angle = -1.3 * (pi/180), origin = (0, 0, 0), axis = (0, 1, 0))
Saturn.rotate(angle = -2.5 * (pi/180), origin = (0, 0, 0), axis = (0, 1, 0))
Uranus.rotate(angle = -0.8 * (pi/180), origin = (0, 0, 0), axis = (0, 1, 0))
Neptune.rotate(angle = -1.8 * (pi/180), origin = (0, 0, 0), axis = (0, 1, 0))

# Rotate Saturn's rings for cosmetic purposes
SaturnRing.rotate(angle = -45 * (pi/180), origin = (0, 0, 0), axis = (0, 1, 0))

# Initialize new variables
JupiterRing.origin = Jupiter
SaturnRing.origin = Saturn
UranusRing.origin = Uranus
NeptuneRing.origin = Neptune

#Masses are in kg
Sun.m = 1.989e30
Mercury.m = 0.330e24
Venus.m = 4.87e24
Earth.m = 5.972e24
Moon.m = 0.073e24
Mars.m = 0.642e24
Jupiter.m = 1898e24
Saturn.m = 568e24
Uranus.m = 86.8e24
Neptune.m = 102e24

'''
p = m * v
    p = Momentum
    m = Mass
    v = velocity
    Note: All velocities are given and they are the Oribital Velocities of each planet in m/s. Negatives are used to make
the planets go clockwise
'''
Sun.p = (0, 0, 0)
Mercury.p = Mercury.m * vector(0, -4.74e4, 0)
Venus.p = Venus.m * vector(0, -3.5e4, 0)
Earth.p = Earth.m * vector(0, -2.98e4, 0)
Moon.p = Moon.m * vector(0, -(2.98e4 + 1e3), 0) # Sum of the Earth and Moon's Orbital Velocities
Mars.p = Mars.m * vector(0, -2.41e4, 0)
Jupiter.p = Jupiter.m * vector(0, -1.31e4, 0)
Saturn.p = Saturn.m * vector(0, -9.7e3, 0)
Uranus.p = Uranus.m * vector(0, -6.8e3, 0)
Neptune.p = Neptune.m * vector(0, -5.4e3, 0)

deltaTime = 24 * 60 * 60 # The change in time. Note that: 24 hours/day * 60 mins/hour * 60 seconds/min = seconds in a day

# Turn trails back on after the scene has finished setup
for planet in [Mercury, Venus, Earth, Moon, Mars, Jupiter, Saturn, Uranus, Neptune]:
    planet.make_trail = True

'''
F = -G * M * m * rhat/r^2
    F = Force of Gravity
    G = The Gravitational Constant, 6.674 * 10^-11 m^3 * kg^-1 * s^-2
    M = Mass of the Sun
    m = Mass of the planet
    r = The distance between the Planet and the Sun
    rhat =  The normal vector between the Planet and the Sun
```

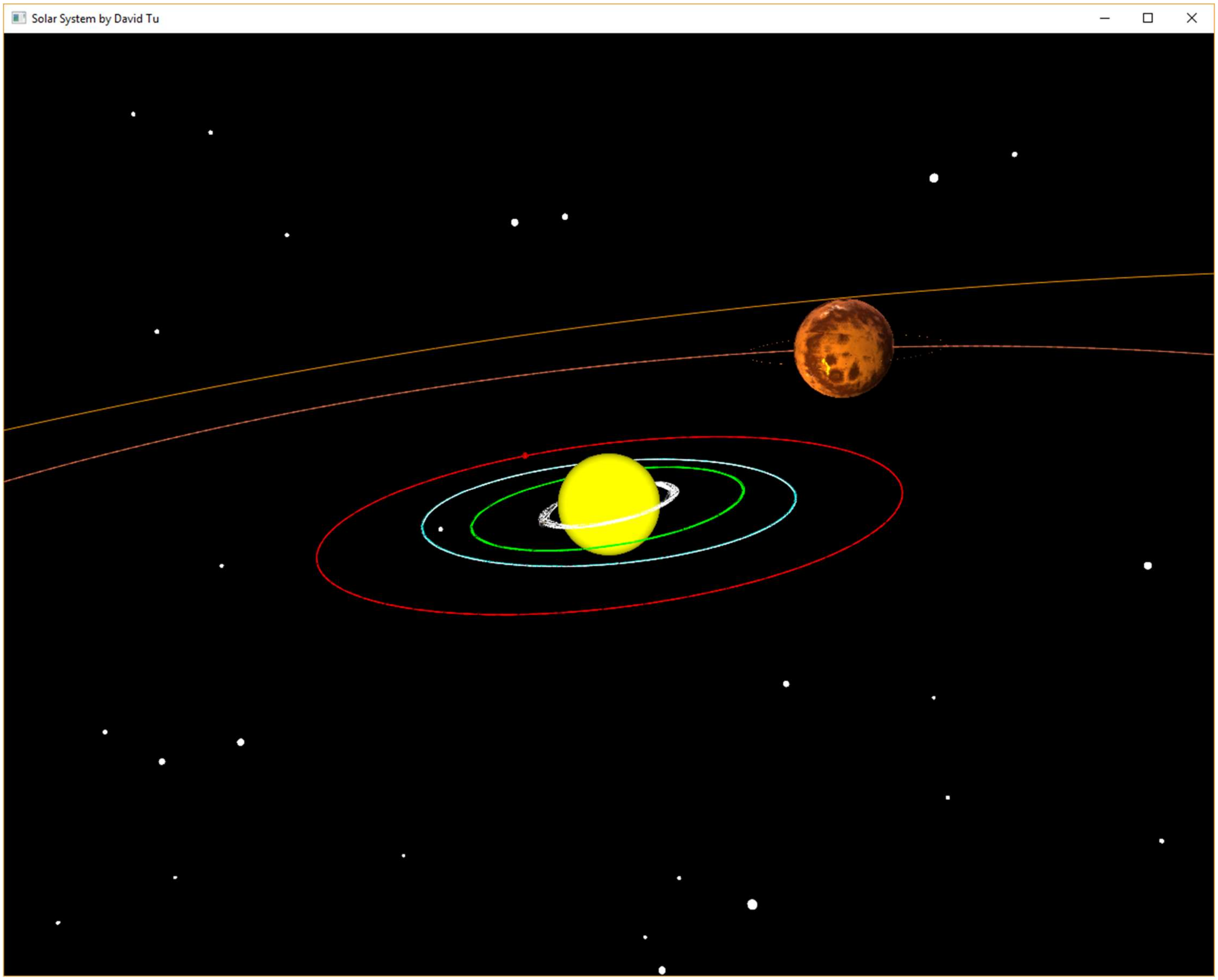
```
            The normal vector can be calculated by taking the vector between the Planet and the Sun and dividing it by
it's magnitude
'''
while True:
    distance = Mercury.pos - Sun.pos # Note that this is a vector pointing from the Sun TO the planet
    force = -(G * Sun.m * Mercury.m * distance / mag(distance)**3)# The distance between the Sun and the Planet can be
                                                                  # obtained by calculating the magnitude
    Mercury.p += force * deltaTime # Update momentum

    distance = Venus.pos - Sun.pos
    force = -(G * Sun.m * Venus.m * distance / mag(distance)**3)
    Venus.p += force * deltaTime

    distance = Earth.pos - Sun.pos
    force = -(G * Sun.m * Earth.m * distance / mag(distance)**3)
    Earth.p += force * deltaTime

    distance = Moon.pos - Sun.pos
    distanceFromEarth = Moon.pos - Earth.pos
    forceOfSun = -(G * Sun.m * Moon.m * distance / mag(distance)**3)
    forceOfEarth = -(G * Earth.m * Moon.m * distanceFromEarth / mag(distanceFromEarth)**3)
    force = forceOfSun + forceOfEarth
    Moon.p += force * deltaTime

    distance = Mars.pos - Sun.pos
    force = -(G * Sun.m * Mars.m * distance / mag(distance)**3)
    Mars.p += force * deltaTime

    distance = Jupiter.pos - Sun.pos
    force = -(G * Sun.m * Jupiter.m * distance / mag(distance)**3)
    Jupiter.p += force * deltaTime

    distance = Saturn.pos - Sun.pos
    force = -(G * Sun.m * Saturn.m * distance / mag(distance)**3)
    Saturn.p += force * deltaTime

    distance = Uranus.pos - Sun.pos
    force = -(G * Sun.m * Uranus.m * distance / mag(distance)**3)
    Uranus.p += force * deltaTime

    distance = Neptune.pos - Sun.pos
    force = -(G * Sun.m * Neptune.m * distance / mag(distance)**3)
    Neptune.p += force * deltaTime

    for planet in [Mercury, Venus, Earth, Moon, Mars, Jupiter, Saturn, Uranus, Neptune]:
        planet.pos += planet.p / planet.m * deltaTime #Update the position of the Planet with velocity (v = p / m)

    for ring in [JupiterRing, SaturnRing, UranusRing, NeptuneRing]:
        ring.pos = ring.origin.pos # Update the position of the rings

    if EarthMoonView:
        scene.center = Earth.pos
        rate(20) # Framerate
    else: rate(200)
```

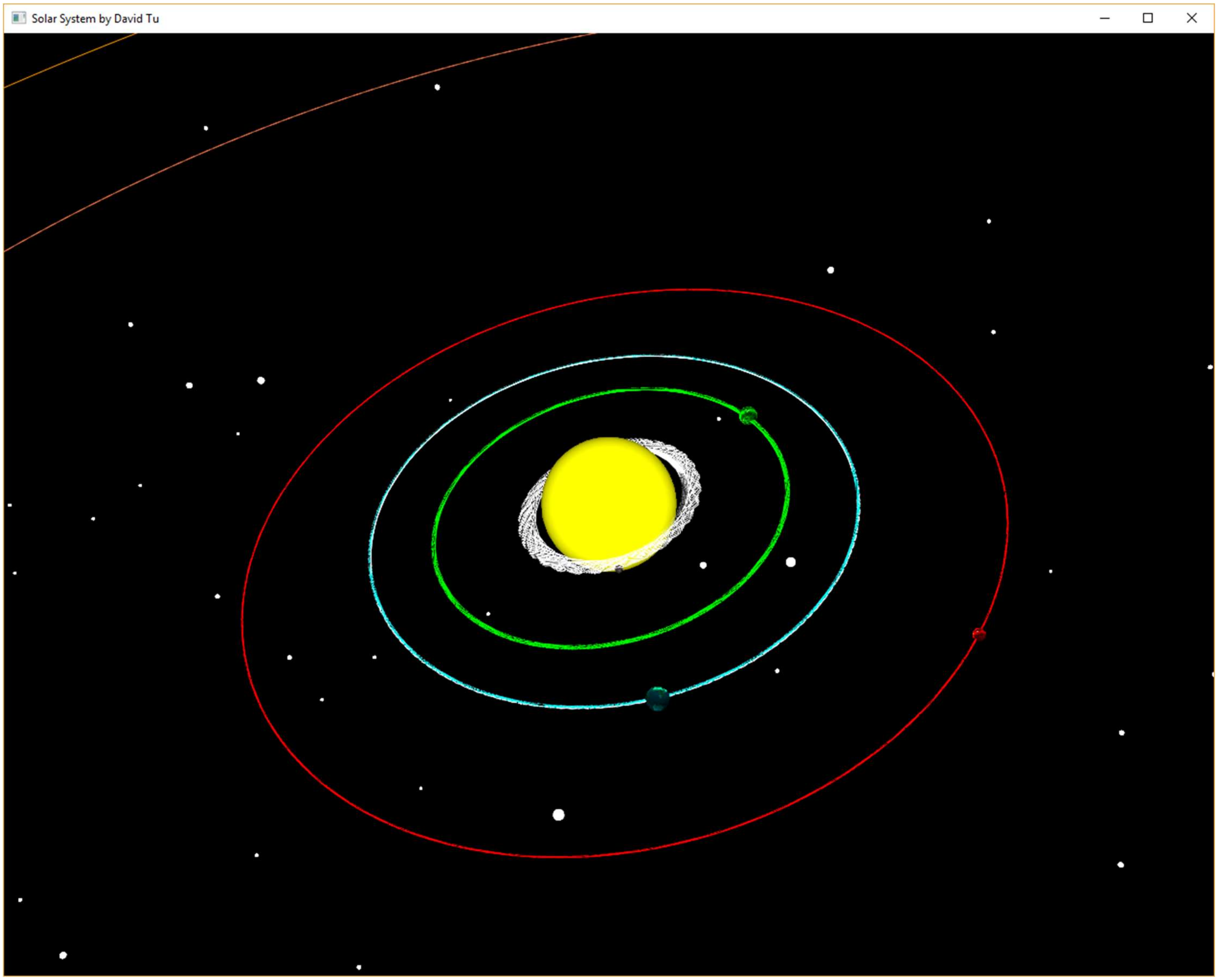
Screenshots

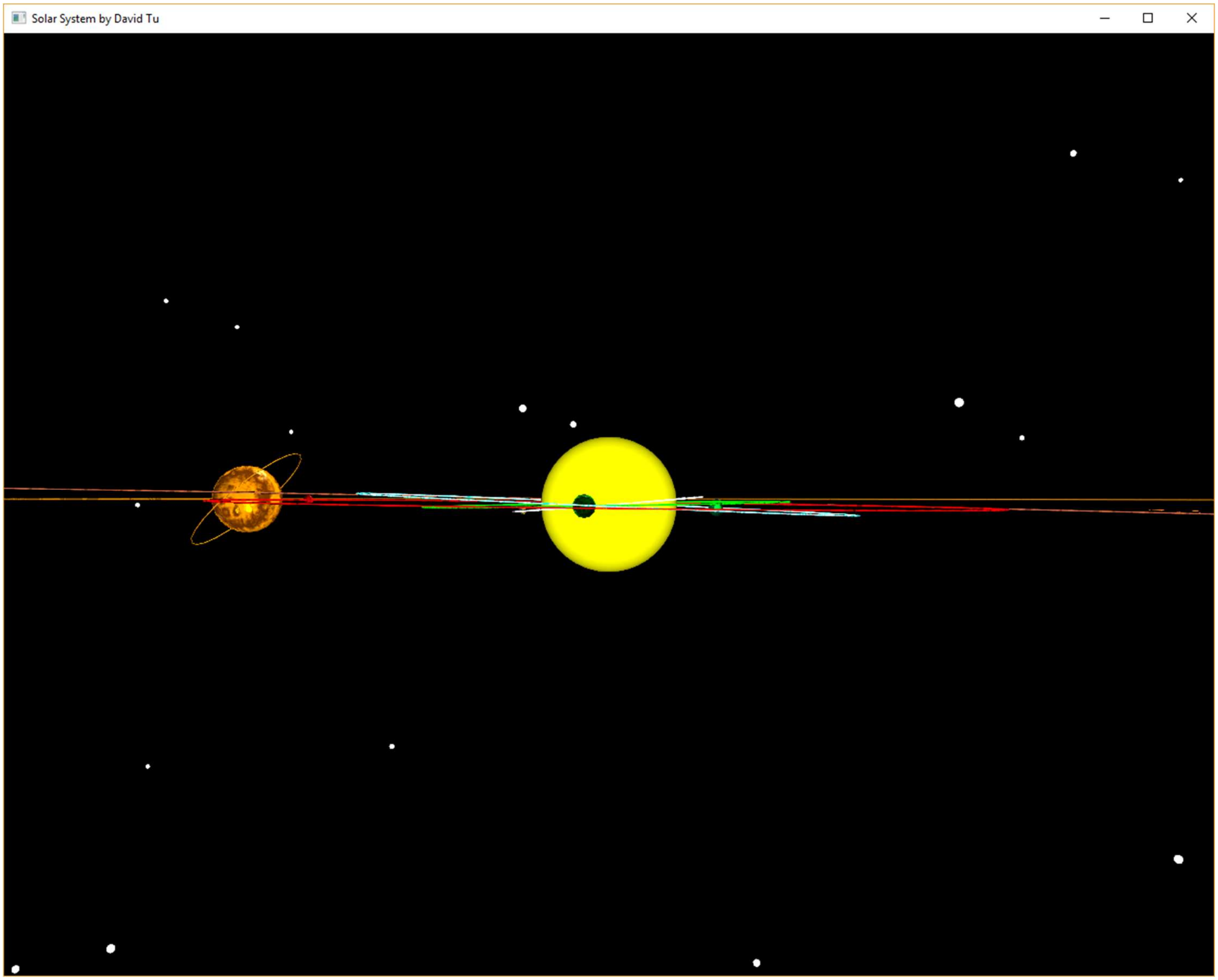
Scaled Solar System View (When “EarthMoonView = False”):

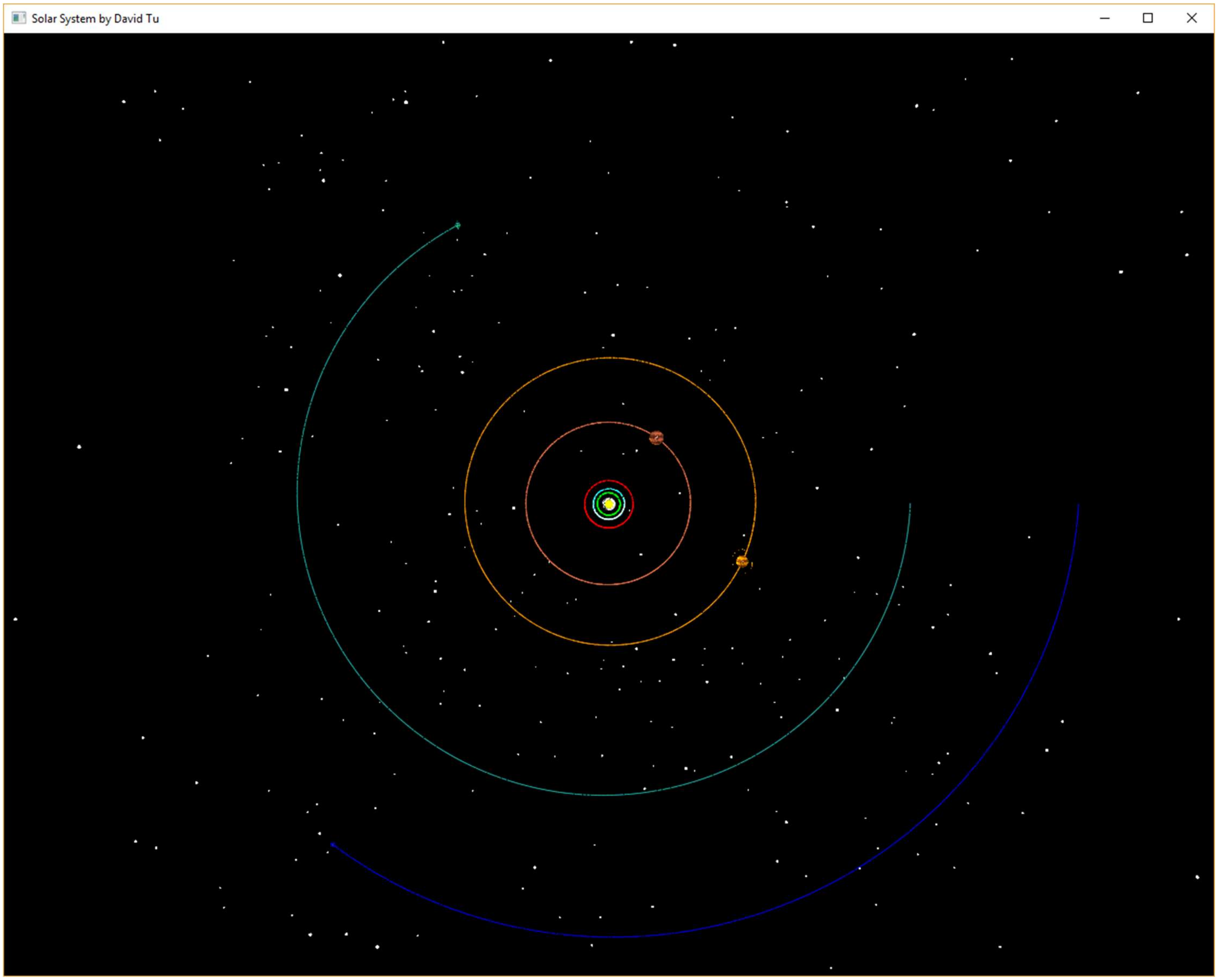


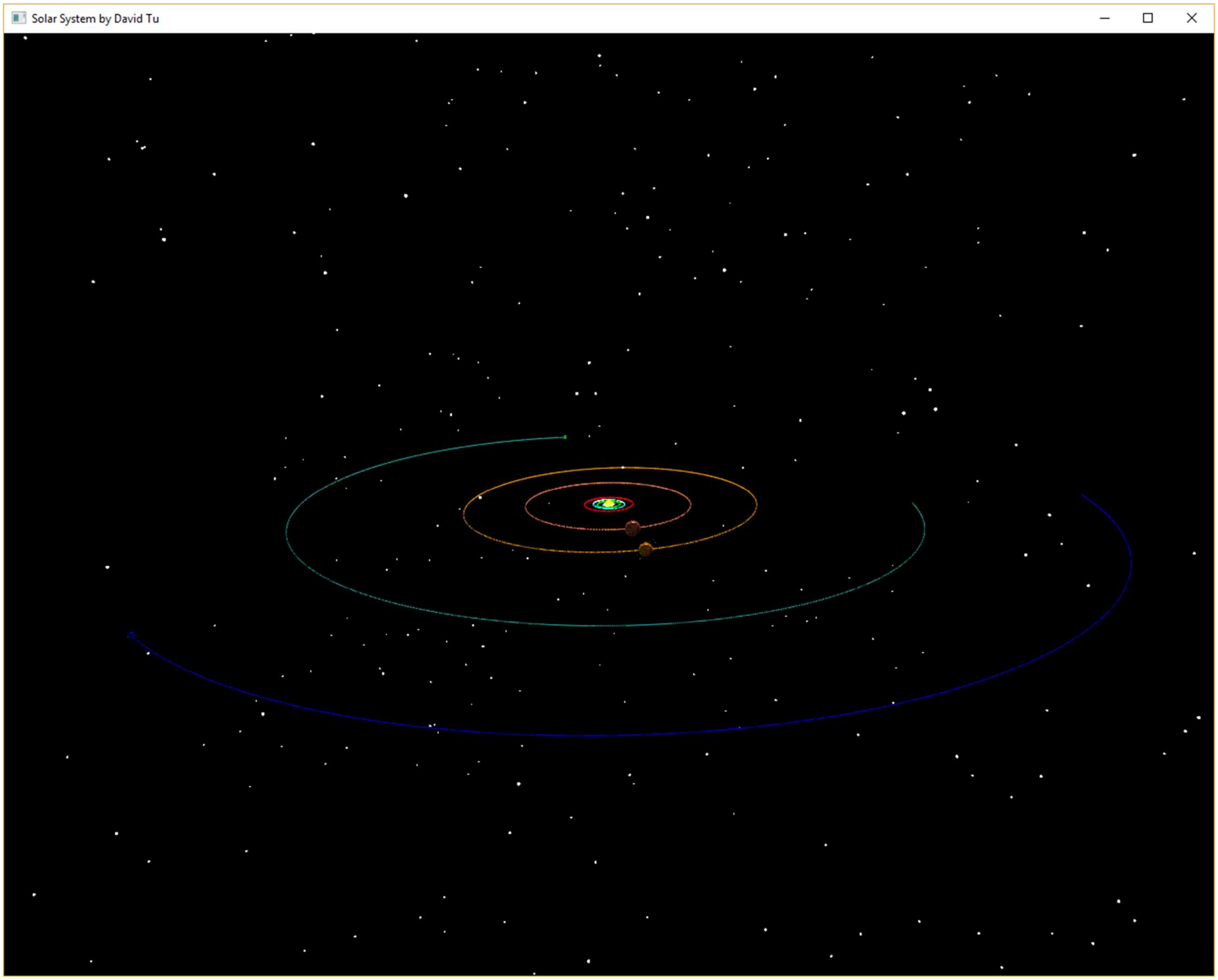

Solar System by David Tu

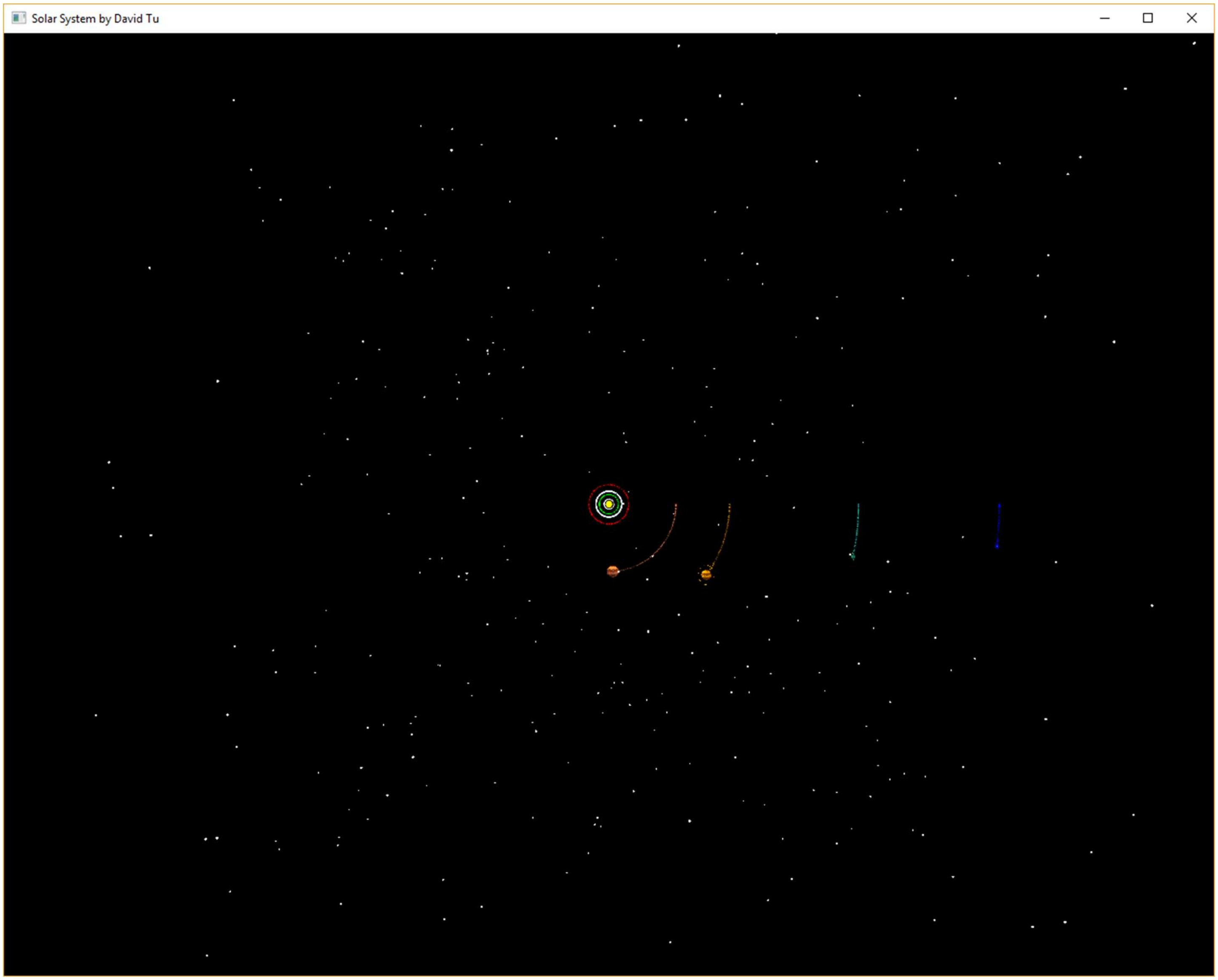

Solar System by David Tu


Solar System by David Tu

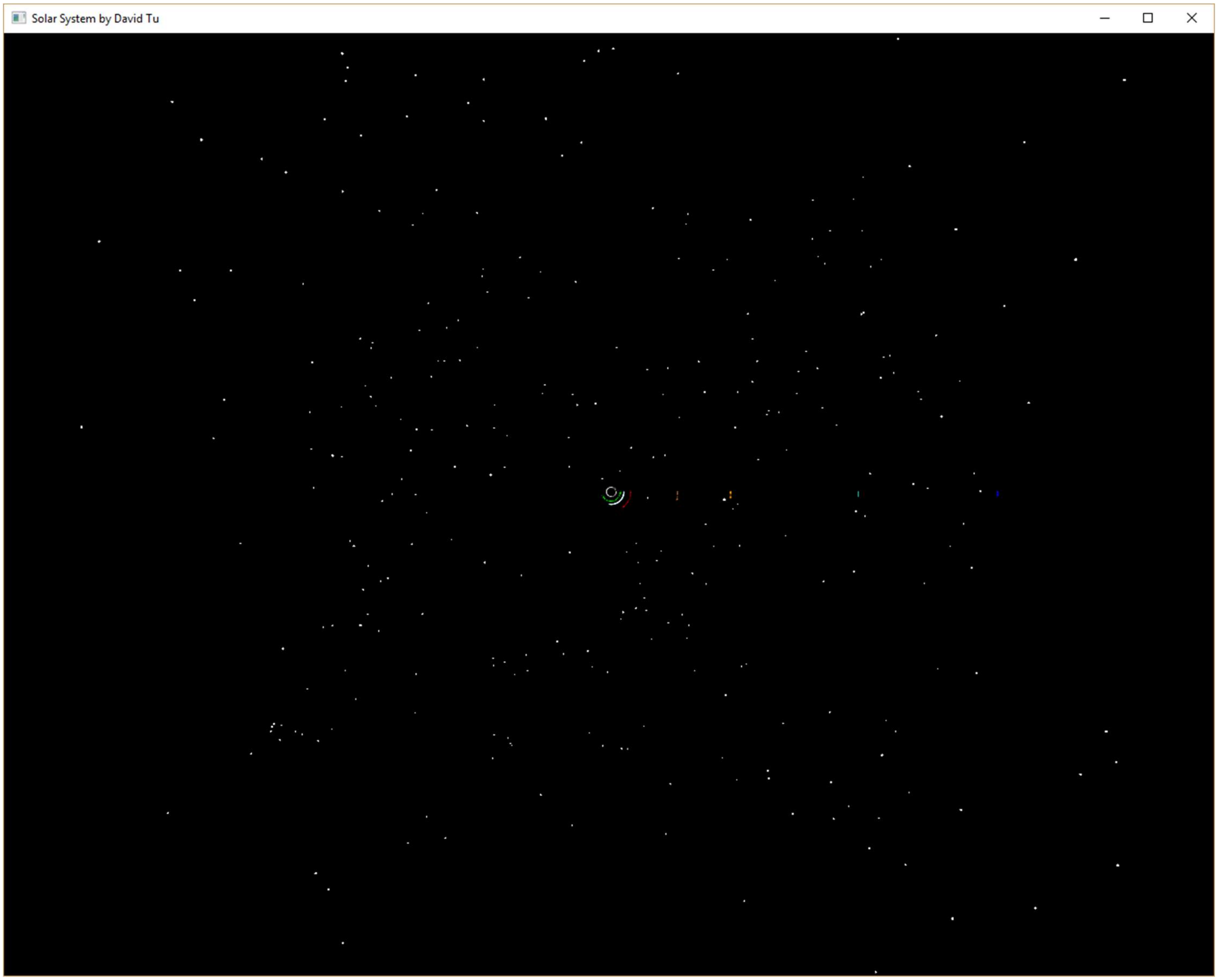

Solar System by David Tu


Solar System by David Tu

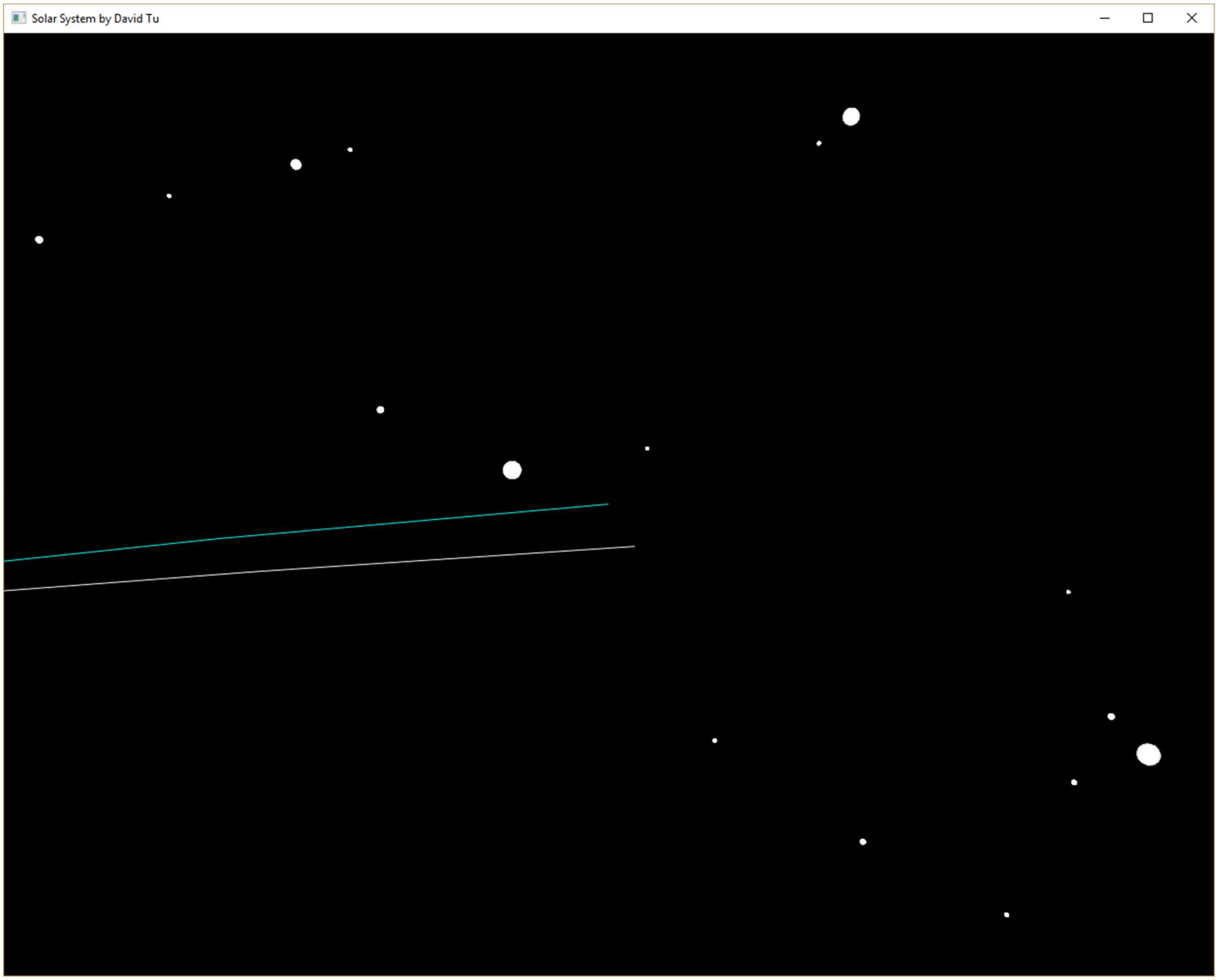

Solar System by David Tu

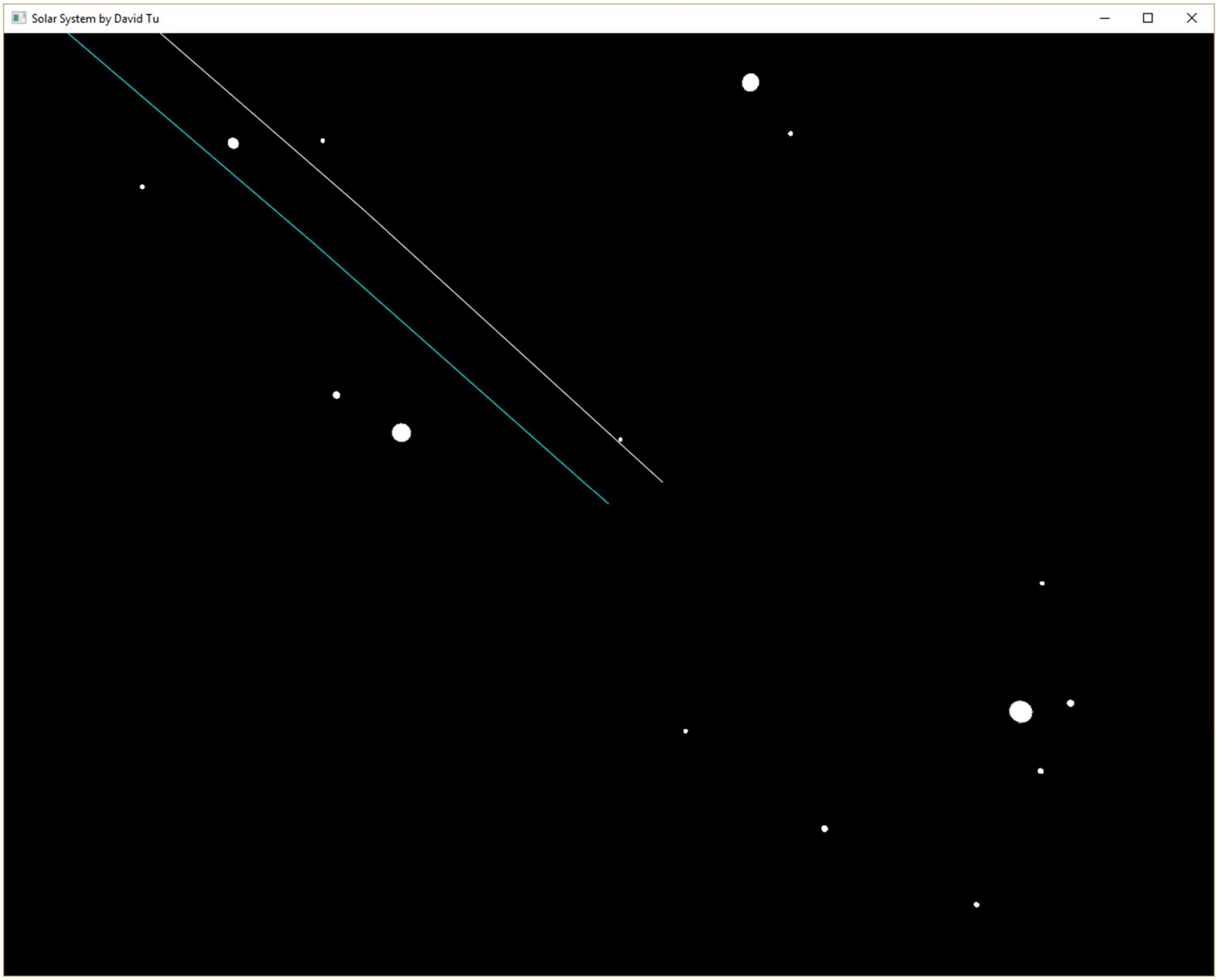

Solar System by David Tu

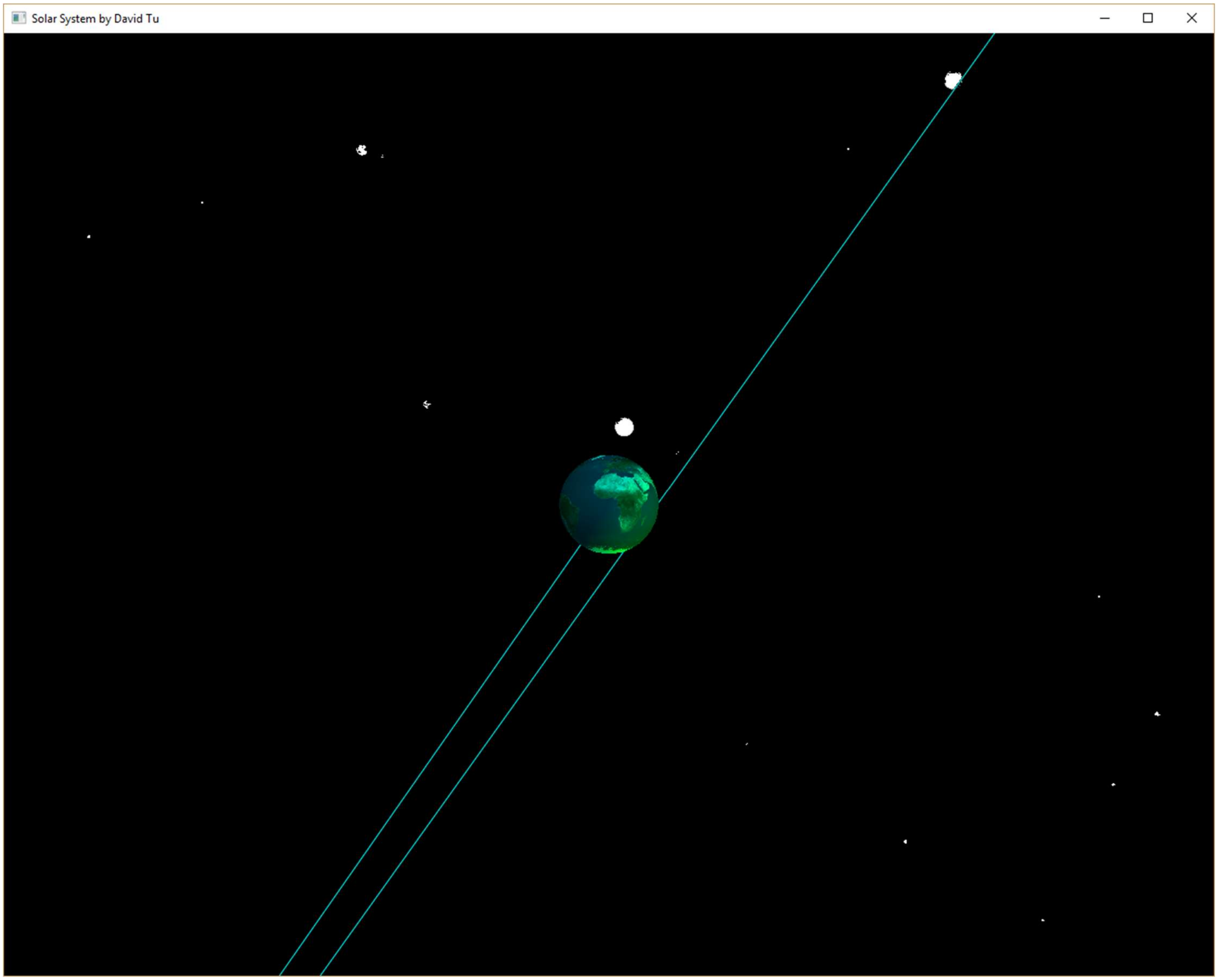
Earth-Moon View - the Solar System, unscaled (When “EarthMoonView = True”):

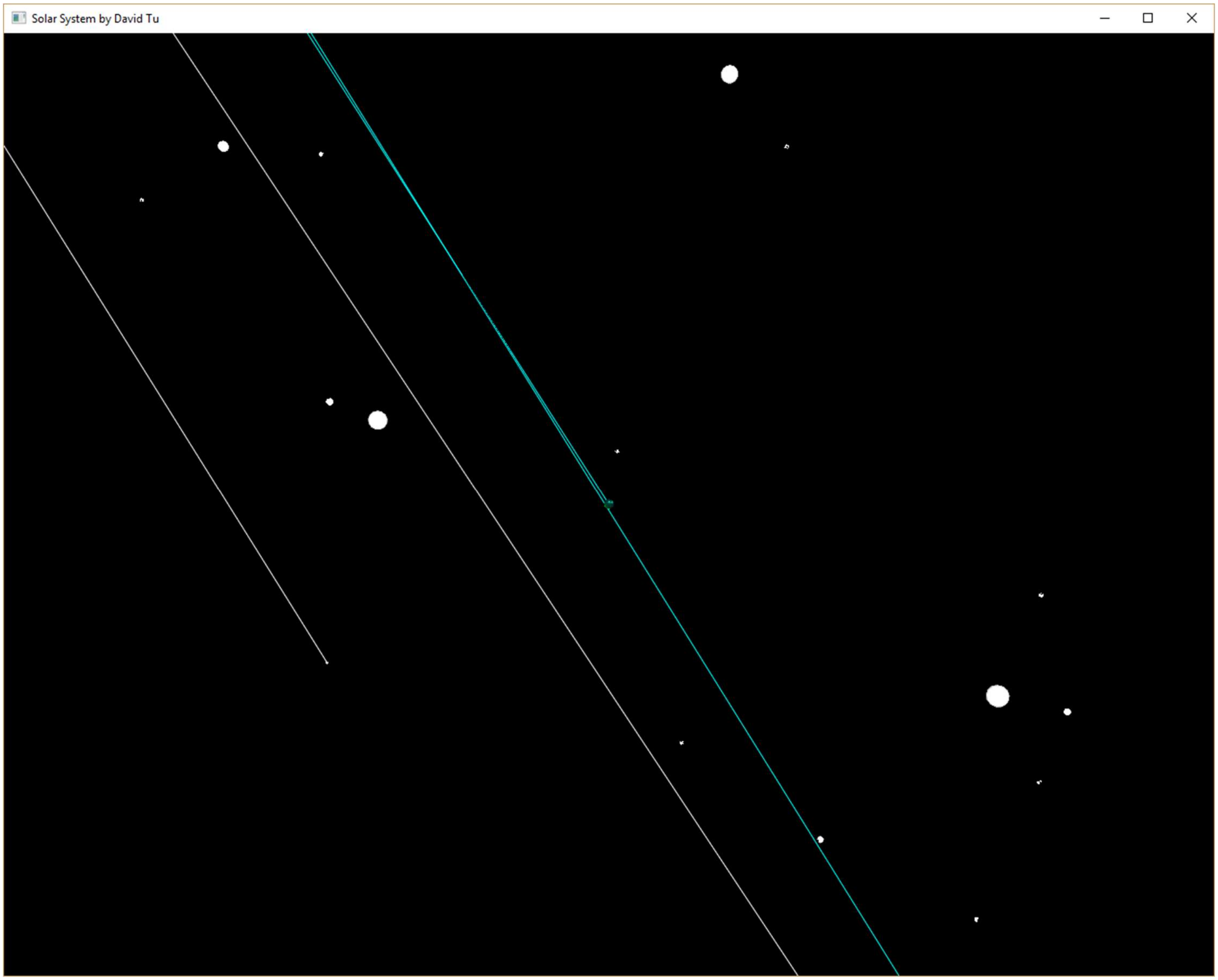


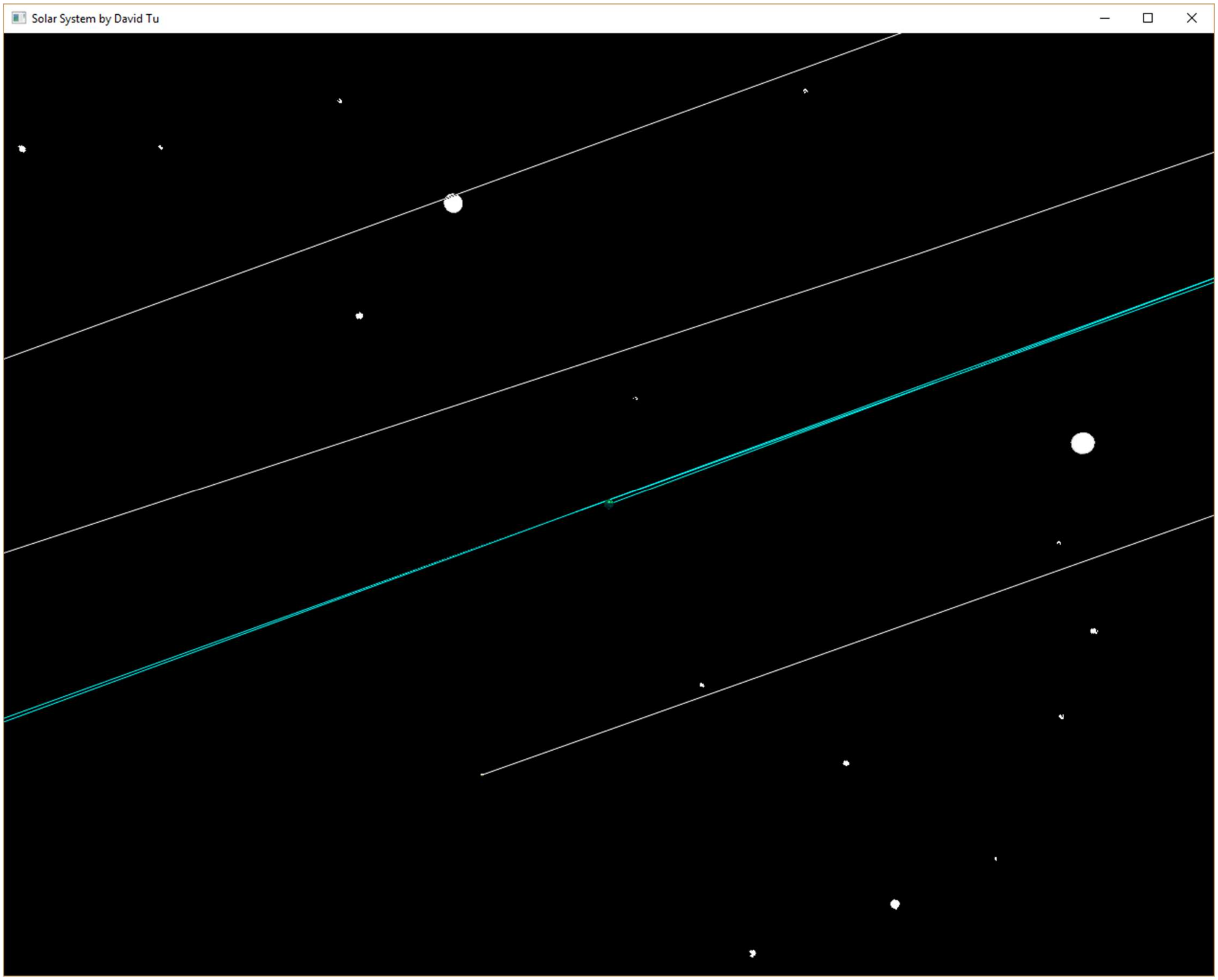
When zooming in on the Earth, you'll be able to see the Moon orbiting around the Earth:

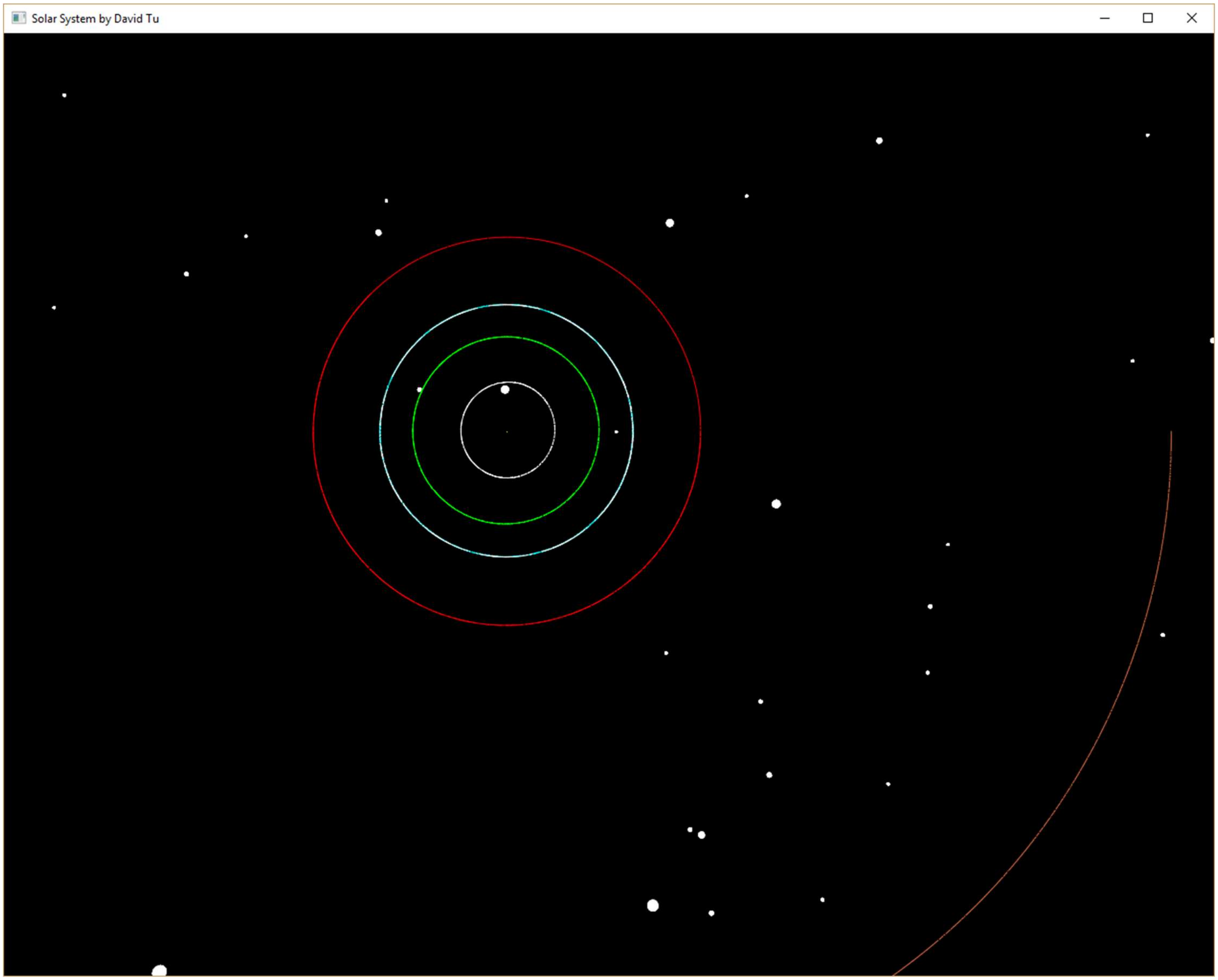


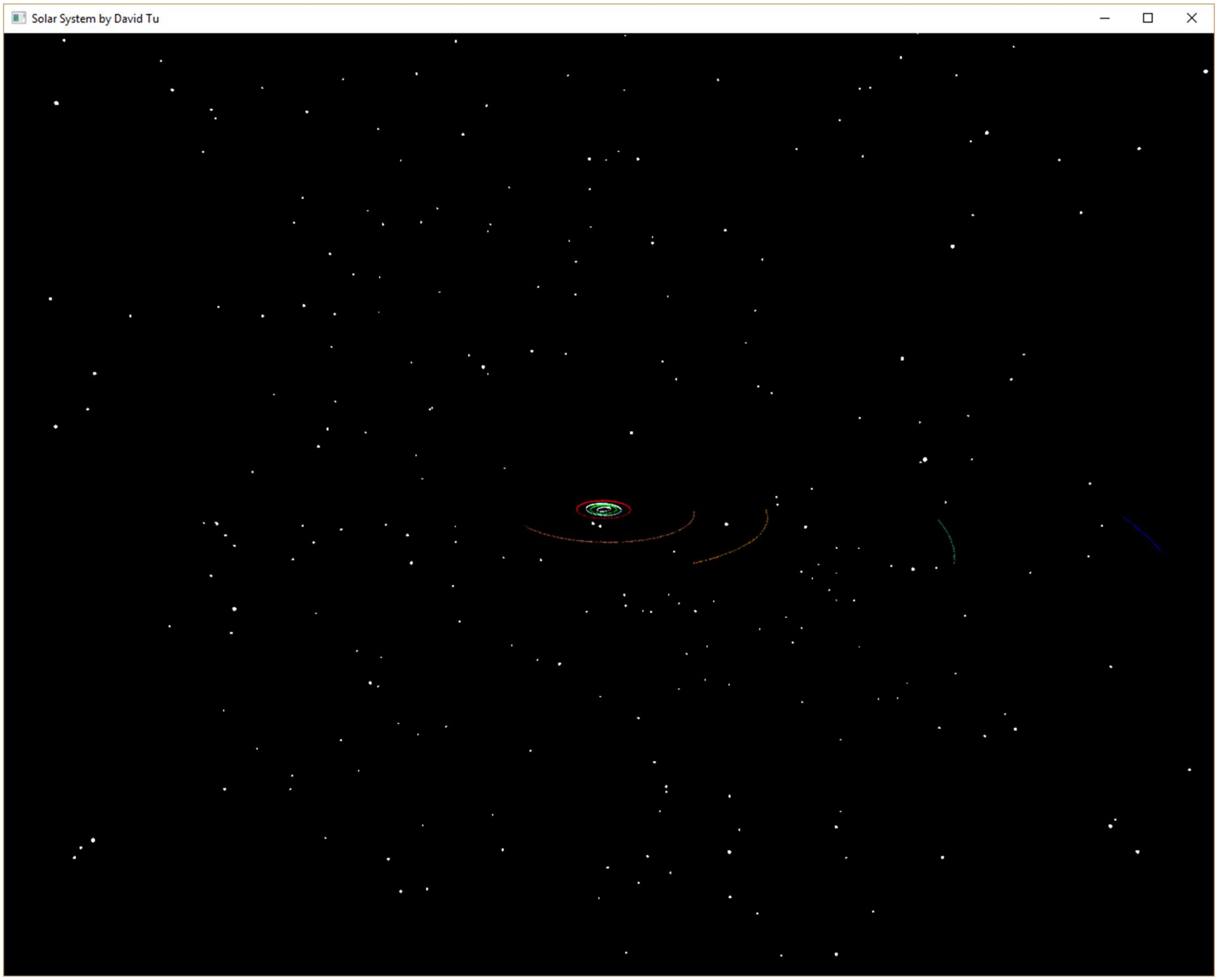

Solar System by David Tu


Solar System by David Tu


Solar System by David Tu


Solar System by David Tu


Solar System by David Tu


Solar System by David Tu