Solar System Scale Model

someonesdad1@gmail.com 16 Sep 2011

Many of us learn about the solar system in elementary science classes. But we don't go much beyond learning a few facts and perhaps looking at a few of the planets in the night sky. It's instructive to make a scale model of the solar system to get a feel for how big the thing really is. I think most people will be surprised at how much space is between the planets and how small the planets are when you look at a scale model -- I certainly was.

You can model the solar system by pounding some wooden stakes in the ground at the proper distance from the simulated Sun. You'll probably want to do this in a field where you have 100 m to 200 m unobstructed. If you want the Sun to be the size of a basketball (75 cm circumference or 239 mm diameter), then you'll need to put the simulated Neptune about 770 m (or nearly half a mile) away.

To help constructing a model, I wrote a python script that performs the calculations. It doesn't do anything you can't do with a pencil and a piece of paper (or a spreadsheet); it just helps save a bit of time. If you don't have python, you can get it from http://www.python.org/. Install it, then run the script from the command line as follows: `python solar_system_plot.py`. You'll have to include a number on the command line; this number is the distance in meters over which the solar system model will extend (the script will give you a help message). The script is at the end of this paper.

The default script prints out a table that includes the Sun and the planets from Mercury to Neptune. You can edit the script to e.g. include some of the minor planets.

I constructed a model in my back yard. The first thing I needed to do was to measure the usable space. I wanted a "straight shot" along the grass; the longest I could come up with was 90 m (when it got darker, I measured the actual distance with my laser distance meter and it was 87.66 m, so my scale model is off a few percent). I ran the script with 90 as the command line argument and got the following output:

```
Model distance = 90  m
Scale factor = 1.99847e-11
```

Body	Diameter mm	Distance m	Perihelion m	Aphelion m	Mass (Earth = 1)	Sun Angle mdegree
Sun	27.8	0	0	0	3.33e+05	
Mercury	0.0975	1.16	0.919	1.4	0.0553	1377
Venus	0.242	2.16	2.15	2.18	0.815	737
Earth	0.255	2.99	2.94	3.04	1	533
Mars	0.135	4.56	4.13	4.98	0.107	349
Jupiter	2.79	15.6	14.8	16.3	318	102
Saturn	2.29	28.6	27.1	30.2	95.2	55.6
Uranus	1.01	57.5	54.9	60	14.5	27.7
Neptune	0.981	90	89	91	17.1	17.7

The Diameter column gives the scaled mean diameter of the body in mm. The Distance column gives the distance from the simulated Sun; this is the semimajor axis of the ellipse of the orbit. The semimajor axis is half of the longest diameter of the ellipse. The Perihelion and Aphelion columns give you the scaled distance of closest approach and farthest distance from the Sun (if you wonder why the aphelion is bigger than the semimajor axis, remember the Sun is at one of the foci of the elliptical orbit). The Mass column gives the mass of the object in units of the Earth's mass. The Sun Angle is the subtended angle of the Sun's disk at the given semimajor axis distance. This is partially a check on the calculation, as it's well-known that the Sun and Moon both subtend about half a degree of arc from Earth. Note the angle is given in millidegrees.

Here's a manual check on the numbers. The semimajor axis of Neptune is 30.10 astronomical units. One astronomical unit is 1.496×10^{11} m, so Neptune's semimajor axis is 4.503×10^{12} m. The scaling

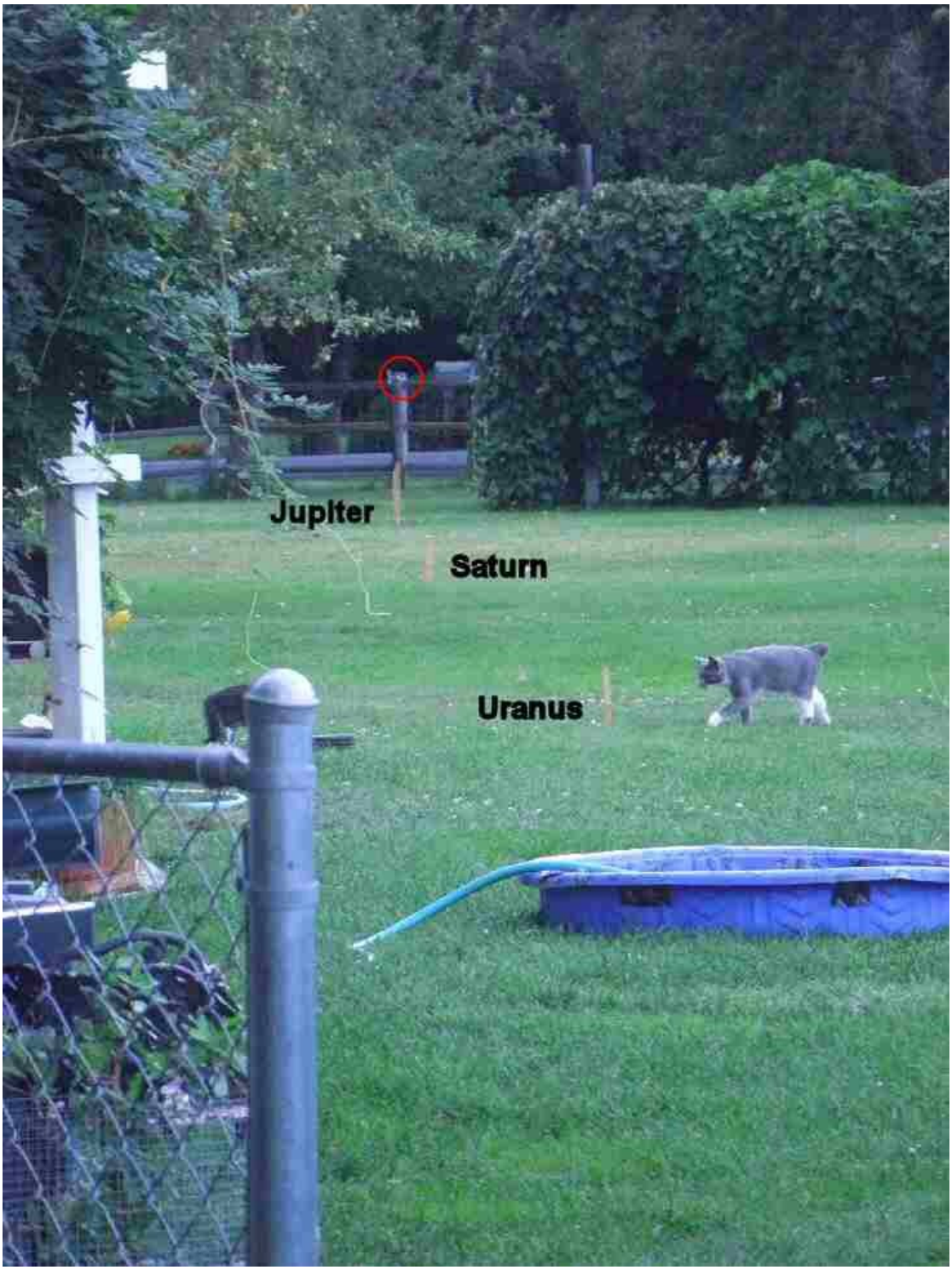
factor to make this equal to 90 m is $90/4.503\times10^{12}$ or 1.9987×10^{-11}. Jupiter's mean radius is 6.9911×10^{7} m; multiplying this by the scaling factor yields 0.001397. Doubling this to get diameter and then multiplying by 1000 to get mm gives 2.794 mm; 2.79 is given in the table.

I measured the mass of a Daisy BB (a steel sphere 4.3 mm in diameter) at 0.6 g (my Ohaus mechanical scale has a resolution of 0.1 g). Suppose this were the mass of Earth. Then the mass of the Sun would be 200 kg -- roughly the mass of three people. Jupiter at this scale has a mass of 200 g -- about the mass of water filling a typical coffee cup.

For this scale model, the Sun is about 2/3 the diameter of a golf ball. The inner planets have diameters of about 0.1 to 0.2 mm -- to put this in perspective, this is from 38 gauge to 32 gauge in the American Wire Gauge size. If you've worked with electrical stuff, you'll know that those are pretty small wires. You'd have a hard time seeing those dots from 10 meters away -- this tells you that if there was no reflected light from the planet, you'd never see it at interplanetary distances (unless it occluded a star or transited the Sun).

For the simulated Sun, I cut out a 28 mm circle from white cardstock and pinned it on the fencepost 90 m from where Neptune would be. I couldn't even see the disk of the Sun without using 7x50 binoculars from the Neptune distance. This was on a sunny day, although the disk was in the shade.

Here's a picture of the view from Neptune at the maximum zoom (10X) of my camera:



The distance from where I'm standing to the stake marked Uranus is 32.5 m. The cats help give scale (there's also a cat behind the post). The Sun is inside the red circle on the gray fence post in the distance behind the stake at Jupiter. Here's a 10X zoomed in picture taken from Saturn:

The zoomed-in inset picture shows the 28 mm diameter disk that represents the Sun (the stake representing Saturn's distance is large and out of focus). This picture was taken at a little over 28.6 m from the post the Sun is on.

Each of the stakes for the planets has a small piece of cardstock on it with the planet's scaled size drawn on it. For the four inner planets, I wasn't able to make a dot on the paper small enough (the dots measured around 0.3 to 0.4 mm -- but they needed to be 0.1 to 0.2 mm). It's fairly humbling to walk to Jupiter and see this planet, the largest in our solar system (11 times Earth's diameter), as a 2 mm diameter circle at this scale -- about half the diameter of a BB.

Travel is limited by the speed of light. It takes 4.2 hours for the light from the Sun to reach Neptune. To simulate this speed, you'd need to walk at about 360 mm per minute. That's about the length from the tip of your fingers to your elbow joint every minute.

The nearest star Proxima Centauri is about 4.2 light years away. In this scale model, that star would be **800 km** away. So from the fence post with the Sun in this model, Neptune is 90 m away (I used to be able to throw a rock that far), but the nearest star to Earth is one to two states away (I live in the western US). That's a lot of empty space. The 100,000 light year diameter of the Milky Way galaxy (at this scale, mind you), is one light-minute away (or 47 times the distance to the Moon). Are you starting to get agoraphobic? ☺

I strongly recommend you take the time to make such a layout -- it will give you a sense of the scale of the solar system that no textbook or numbers can. Even better, do it with your kids or grandkids -- it's something they'll remember.

Here's a video that discusses this topic: http://wimp.com/distanceplanets/.

OK, one more factoid tidbit. The sun puts out energy at the rate of 3.85×10^{26} W. That's a lot of power. What fraction of this does the Earth get? Earth's mean distance from the sun is 1.5×10^{11} m and Earth's diameter is 1.27×10^{7} m. The angle subtended by the Earth from the sun is thus

$$\theta = 2\tan^{-1}\frac{1.27\times10^{7}}{2(1.5\times10^{11})}$$

This is 42.3 μradians. This translates into the solid angle of $2\pi(1-\cos\theta)$ or 5.63×10^{-9} steradians. Assuming the sun radiates isotropically, divide this by 4π to get the fraction of the sun's energy that reaches Earth: 4.5×10^{-10}. That's still 1.7×10^{17} W, a large amount of power.

A good way to scale this is to relate it to the area of your country. The area of the contiguous US is about 7.7×10^{12} m^2. The fraction of this represented by the fraction of the sun's energy that reaches Earth is 3465 m^2. That's a square 59 m on a side, or an area of 0.86 acres -- which is just a bit under the size of the lot my house is on. Since it takes a number of hours in a plane to fly across the US, this gives you another feeling for the scale of things. Of course, the vast majority of all that energy is lost to interstellar space.

The script

```
'''
Prints out the diameters and distances needed to lay out a scale model of
the solar system.  You give the program the linear distance in meters you
want the model to extend over; then the relative diameters in mm and orbit
distances in m are printed out.

The dimensions in the bodies dictionary were gotten from wikipedia on 15
Sep 2011.

Note that you can add objects to the bodies dictionary.  I've put in the
Sun and the traditional planets, but there are other objects commented out
that you might want to include.

The script goes through the bodies dictionary and finds the object with
the largest semi-major axis for the orbit and scales things so that it's
orbit is at the desired distance in the scale model.

--------------------------------------------------------------------------
Copyright (C) 2011 Don Peterson
Contact:  gmail.com@someonesdad1

                          The Wide Open License (WOL)

Permission to use, copy, modify, distribute and sell this software and its
documentation for any purpose is hereby granted without fee, provided that
the above copyright notice and this license appear in all copies.
THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY OF
ANY KIND. See http://www.dspguru.com/wide-open-license for more
information.
'''

from __future__ import division
import sys, getopt, math
from pdb import set_trace as xx

# The following constant comes from the only argument on the command line
# and represents how many meters are in the semi-major axis of the most
# distant object in the model.
model_distance = None
km2m = 1000
omk2m = 1e9
au2m = 1.495978707e11     # AU to meters

bodies = {
    # Each element contains:
    #   0   Body's mean radius in m
    #   1   Mass in kg
    #   2   Semi-major axis of orbit in AU
    #   3   Perihelion in AU
    #   4   Aphelion in AU
    "Sun"     :
        (1.392e9/2,  1.9891e30,  0, 0, 0),
    "Mercury" :
        (2439.7e3,   3.3022e23,  0.387098, 0.307499, 0.466697),
    "Venus" :
        (6051.8e3,   4.8685e24,  0.723332, 0.71843270, 0.72823128),
    "Earth" :
        (6371e3,     5.9736e24,  1.00000261, 0.98329134, 1.01671388),
    "Mars" :
        (3386.2e3,   6.4185e23,  1.523679, 1.381497, 1.665861),
    "Jupiter" :
        (69911e3,    1.8986e27,  5.204267, 4.950429, 5.458104),
    "Saturn"  :
        (57316e3,    5.6846e26,  9.58201720, 9.04807635, 10.11595804),
```

```
    "Uranus"  :
        (25266e3,    8.6810e25,  19.22941195, 18.37551863, 20.08330526),
    "Neptune" :
        (24552e3,    1.0243e26,  30.10366151, 29.76607095, 30.44125206),
    "Pluto" :
        (1153e3,     1.305e22,   39.264, 29.657, 48.871),
    "Haumea" :
        (718e3,      4.006e21,   43.132, 34.721, 51.544),
    "Makemake" :
        (710e3,      36e21,      45.791, 38.509, 53.074),
    "Eris" :
        (1163e3,     1.67e22,    67.67, 37.77, 97.56),
    "Sedna" :
        (1400e3,     3e21,       518.57, 76.361, 937),
}

earth_mass = bodies["Earth"][1]

# This list determines which bodies to use in the model and their order
order = (
    "Sun",
    "Mercury",
    "Venus",
    "Earth",
    "Mars",
    "Jupiter",
    "Saturn",
    "Uranus",
    "Neptune",
   #"Pluto",
   #"Haumea",
   #"Makemake",
   #"Eris",
   #"Sedna",
)

manual = '''
{name}  model_scale_in_m
  Prints a table showing scaled sizes of the solar system.  You pass in a
  distance in m on the command line and the program fits the model into
  that dimension.  Then it gives the scaled diameters and distances that
  you must put the scaled solar system bodies at.

  Note:  diameters will probably be smaller than you expect and the
  distances involved will be larger than you expect!

  Example:
    I want to fit a solar system model into my back yard.  I have measured
    a suitable 90 m section over which I can put the model.  I call the
    program as follows to print out the scale model's dimensions:

        {name} 90
'''[1:-1]

def Usage():
    name = sys.argv[0]
    print manual.format(**locals())
    exit(1)

def GetLargestDistance():
    '''Find the largest semimajor axis in the bodies dictionary and return
    it in m.
    '''
    largest = 0
    for body in order:
        largest = max(bodies[body][2], largest)
    return largest*au2m  # Distances are in AU, so convert to m
```

```
def SunAngle(semimajor_axis_m):
    '''Return the Sun's subtended angle in millidegrees.
    '''
    sun_radius_m = bodies["Sun"][0]
    angle = 2*math.atan(sun_radius_m/semimajor_axis_m)
    return 1e3*angle*180/math.pi

def main():
    d = {}
    if len(sys.argv) != 2:
        Usage()
    model_distance = sys.argv[1]
    model_distance_in_m = float(model_distance)
    largest_distance_in_m = GetLargestDistance()
    scale = model_distance_in_m/largest_distance_in_m
    earth_mass = bodies["Earth"][1]
    print "Model distance =", model_distance, " m"
    print "Scale factor = %.5e" % scale
    print
    print '''
            Diameter   Distance    Perihelion   Aphelion     Mass       Sun Angle
  Body         mm          m             m          m       (Earth = 1)  mdegree
--------   --------   --------    ----------   --------   ----------- ---------
'''[1:-1]
    digits = 3
    k = au2m*scale
    for body in order:
        radius, mass, semimajor_axis, perihelion, aphelion = bodies[body]
        print "%-8s" % body,
        print "%10.*g" % (digits, 2.0*radius*scale*1000),
        print "%10.*g" % (digits, semimajor_axis*k),
        print "%13.*g" % (digits, perihelion*k),
        print "%10.*g" % (digits, aphelion*k),
        print "%13.*g" % (digits, mass/earth_mass),
        if semimajor_axis:
            angle = SunAngle(semimajor_axis*au2m)
            if angle >= 100:
                print "%9d" % angle
            else:
                print "%9.*g" % (digits, angle)
        else:
            print
    print

main()
```