Introduction:

NASA has been instructed by congress to send a spacecraft to the moon to clean up the debris derelict rovers, landing gear, etc.) left by the previous Apollo moon landings ending in Apollo 17. The mission has been dubbed Apollo 18. Modeling Motion students at Evergreen have been asked to compute the flight profile for Apollo 18. A Saturn V rocket will deliver the spacecraft into an Earth Parking Orbit (EPO) at an altitude 185km above the surface of Earth, two further boosts of the engine will be required to achieve an orbit around the moon from which the Garbage Collection Module (GCM) can descend to the Lunar surface. The first burn, the Translunar Injection (TLI), will move the spacecraft out of EPO and onto a trajectory taking it within 4 lunar radii of the Moon's surface. The final Lunar Orbit Insertion burn will place the orbiter in position for the GCM to do its job.

Your job will be to compute the following and simulate your results to verify correctness and adjust if necessary:

1. The proper velocity for a circular EPO orbit.
2. The proper engine burn time to add the necessary momentum to intercept the moon.
3. The proper position to begin the TLI burn to intercept the moon.
4. Once within range of the moon, the proper direction and burn time to insert the orbiter in a Lunar orbit.

Procedure:

1. Simulation software has been provided to simulate the Earth/Moon system and the rocket. The file `apollo.py` contains the program which requires the modules in `rocket.py`, `earthmoon.py`, `masssystem.py`, and `orbitformulas.py` which defines a number of useful constants and functions. Copy these into a folder and execute `apollo.py`. You should see a tiny rocket appear above the earth and fall down into it because it has no initial velocity. Zooming Further out a tiny moon can be seen orbiting slowly around the Earth. Earth and moon are shown to scale, although time is accelerated. Normally the moon would take about 28 days to make a complete revolution around the earth. The "tiny" rocket is shown enlarged 10000 times to make it large enough to see. (It appears to be 1,100km tall!). All units in the simulation are in terms of kilograms, meters, and seconds.
2. Place the rocket in a circular Earth Parking Orbit 185km above the surface. The rocket will appear to be virtually skimming the air below. The default orbit when you start the simulation is 2 Earth radii above the surface and therefore much higher. Place the rocket at some position on the parking orbit by assigning a `vector` object to `rocket.pos` and assign the necessary velocity vector to `rocket.v` to put it into orbit. The magnitude of the velocity for a circular orbit of a given radius r above the center of the Earth can be found by setting the force due to gravity equal to centripetal force must balance for a circular orbit

$$\frac{GM_e m}{r^2} = \frac{mv^2}{r}$$

and solving for v since these two forces where $G = 6.672 \times 10^{-11}$ is the universal gravitational constant and $M_e = 5.976 \times 10^{24}$ is the mass of Earth. The mass of the rocket is m but this cancels in this equation since the orbital velocity does not depend on the mass of the rocket.

Simulate to verify your calculations are correct. The rocket should be zipping along very close to the surface of Earth keeping a constant altitude. If you wish you can slow down the simulation by inserting a call to the `rate()` function in the main while loop at the end and passing it the desired frame rate (in frames per second) as an argument.

3. The next step is to try to burn the engine for the right period of time to provide the impulse to move the rocket into a Hohmann transfer orbit–which is an elliptical orbit whose *perigee* is tangent to the correct orbit and whose *apogee* is tangent to the desired orbit (around the moon). The strategy then is to find the necessary velocity at the perigee of this elliptical orbit and burn our engine long enough to give us this velocity. This should insert the rocket the right elliptical orbit whose apogee should be right at the moon's orbit. The necessary velocity can be obtained by solving the energy equation

$$E = \frac{1}{2}mv^2 - \frac{GM_em}{r} = -\frac{GM_em}{2a}$$

for v where a is the semimajor axis of the ellipse given by

$$a = \frac{perigee + apogee}{2}$$

Once we know the proper target velocity we must subtract our existing velocity to find out how much extra velocity needs to be provided by the engine burn. Assume our engine gives a constant thrust of 33.366 million Newtons. Recall that impulse I is the change in momentum produced by a force acting over time. We can use this to compute how long our engine needs to burn to change our velocity the right amount

$$I = \int_{t_i}^{t_f} F dt = F(t_f - t_i) = m(v_f - v_i)$$

We need to solve for $(t_f - t_i)$ to get the burn duration.

Use the duration you have calculated and the given thrust in the call to the rocket's `rocket.burnEngine(`*duration*`,`*thrust*`)` method in the simulator. Your rocket should move in an ellipse that reaches out to the orbit of the moon, but it probably doesn't reach the moon's orbit *at the same time* as the moon.

4. Now you must compute the proper starting position for the perigee of our Hohmann transfer orbit so that we will reach a point on the moon's orbit about the same time as the moon does to actually get within range of the moon's gravitational pull. Kepler's third law gives us the period T of an orbit.

$$T^2 = \frac{4\pi^2 a^3}{GM_e}$$

We can compute our transit time, t to the moon using this equation since we know our path to the moon is one half of a Hohmann orbit. We can also compute the period T of the moon's orbit. Assuming the moon's orbit is nearly circular, we can figure out how much of its full period it travels in radians during our transit time.

$$\theta = 2\pi \frac{t}{T}$$

This angle θ is how much we need to advance the point at which we enter the transfer orbit so we will rendezvous with the moon.

5. the final step is to insert our rocket into a circular Lunar orbit once we reach the moon with the proper engine burn. This is the same as the step needed to insert the rocket into the transfer orbit we are just about to exit from. The program already recognizes when the rocket gets within 5 Lunar radii of the moon and executes a small piece of code which set the orientation of the rockets axis (a vector) and burns the engine one last time. To review, you will need to compute the target velocity, find the difference between that target and the current velocity, compute the required impulse to supply that change in velocity and how long to burn the engine to achieve that impulse. Point the rocket it the right direction and fire the rocket engine. Notice that the target velocity is the vector sum of the moon's orbital velocity and the intended circular orbit velocity.

   Congratulations, you have achieved Lunar orbit! Now you're officially a rocket scientist.

You may find the following predefined variables useful in this program:

- The radius of the Earth Parking Orbit is `EPO`
- The position of the rocket is `rocket.pos`
- The velocity of the rocket is `rocket.v`
- The position of the moon is `system.moon.pos`
- The velocity of the moon is `system.moon.v`
- The radius of the Earth is `Re`
- The mass of the Earth is `Me`
- The radius of the moon is `Rluna`
- The mass of the moon is `Mluna`
- The length of the moon's semimajor axis is `Moon_semimajor_axis`
- The `norm()` function returns a unit vector in the direction of its vector argument.
- The `mag()` function returns the magnitude of its vector argument.

NOTE: You may find it helpful to speed up each trial by increasing the stepsize from one second up to as much as a minute (60 seconds). Remeber larger stepsizes reduce accuracy and too large of a stepsize will cause the simulation to "blow up".