Modeling the Sun-Earth-Moon System

New Mexico
Supercomputing Challenge
Final Report
March 29, 2009

Team 98
Silver High School

Team Members
Powell Brown, Senior
Ted Benakis, Senior
John McCauley, Senior

Teacher
Peggy Larisch

Project Mentors
Berry Estes
Reid Wistort

Acknowledgements

The authors wish to acknowledge and express thanks to the following individuals for the guidance provided in the selection of a suitable topic, review and understanding of results, and assistance provided in the overall preparation of this report:

- Mrs. Peggy Larisch – Teacher, Silver High School, Sponsor
- Mr. Brian Bruessel – Teacher, Silver High School, Physics
- Mrs. Catherine Casey – Teacher, Silver High School, English
- Mr. Berry Estes – Physicist (retired), Sandia National Laboratories
- Mr. Reid Wistort – Senior Engineer, Compilable Memory Arrays, IBM Systems and Technology Group
- Mr. Tom Katonak – Physicist (retired), United States Air Force
- Dr. Elizabeth Kallman – Staff Scientist, Ball Aerospace and Technologies

In Memory of Berry Estes

Abbreviations and Acronyms

- EMCOM – Earth-Moon Center of Mass (Earth-Moon Barycenter) 5
- J2000 Coordinates – Coordinate system based on the elliptic and mean equinox of the reference epoch (January 1, 2000 at 12:00 UTC) 2
- MB – Megabyte 10
- RAM – Random Access Memory 10

List of Figures

Table of Contents

E.0 Executive Summary

At this time, most ephemeris generators available to the public require the user to have a strong understanding of astronomical coordinate systems and the celestial sphere. In general, current ephemeris databases calculate the positions of celestial bodies in a spherical coordinate system. Most people, however, are familiar with the Cartesian coordinate system and find spherical coordinates hard to understand. As such, the current ephemeris databases are difficult to navigate, understand, and apply.

The goal of this project is to create a user-friendly ephemeris generator for the earth and moon with both numerical and visual output. Another goal is to create a stable platform for further investigations of earth-moon-sun interaction. To avoid the instabilities of N-Body simulations and the problems associated with Euler's method, all positional calculations are based on Kepler's Laws. This is achieved through calculating positions at specific intervals using trigonometric functions. The final program is written in DarkBasic, and has an interface layout similar to common software. Because of this, an ephemeris generator has been created that bridges the gap between high powered scientific and amateur investigation, satisfying the need for an accurate generator that is easily utilized and installed on a home computer. At the time of this writing no other ephemeris generators are known to have visual output while providing a stable environment for further research.

1.0 Introduction

1.1 Purpose

Due to the apparent lack of user-friendly ephemeris generating software, this project develops an accurate mathematical model of the sun-earth-moon system and applies it to a program with numerical and visual output. The visual output aides the amateur in understanding the numerical ephemeris data, and also serves as a base for future studies. Current ephemeris software generally outputs in spherical coordinates[1], yet most people are more familiar and comfortable with Cartesian coordinates. As such, all output data is in Cartesian coordinates.

1.2 Scope

To reduce complications, only the sun-earth-moon system is modeled. This assumes that the earth-moon system receives no interference from other bodies in the solar system and farther reaches of space. To further simplify the calculation process, the assumption is made that the sun has infinite mass compared to the earth-moon system. Figure 1.1, courtesy of Dr. Elizabeth Kallman, provides a good visual representation of why this assumption was made. This assumption also allows a perfect heliocentric coordinate system to be created; otherwise, the system would

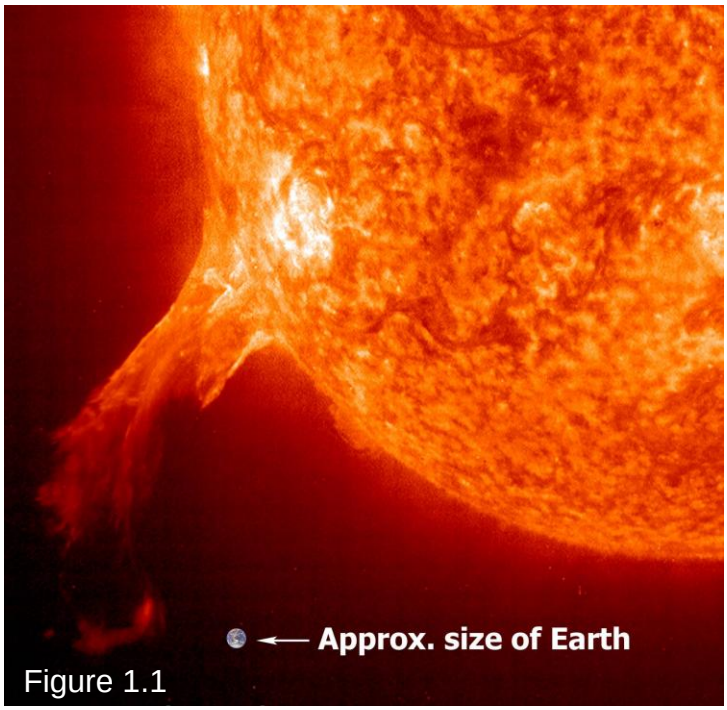


Figure 1.1

originate at the barycenter of the sun-earth-moon system. Baseline calculations will be synchronized with the J2000 coordinate system to easily compare with other ephemeris data.

1.3 Computer Program

The final program is written with the DarkBasic Programming Tool. This language is a modified form of Basic that compiles in C++. It is well suited to this project because it provides good visual modeling along with the ability to perform complex mathematical calculations. The final program has been created from scratch and required the input of all team members. Because of the complex calculations involved with the mathematical model, and the nature of three dimensional computer programs, the final program is the result of work between project mathematicians and programmers.

2.0 Project Proposal

This project develops a three-step process for the creation of an accurate model of the sun-earth-moon system. At this time, there is a need by amateurs and professionals alike for a user-friendly ephemeris generator that can also serve as a baseline for other work. As such, this model shall be functional as an ephemeris generating software and will be able to be applied to other investigations involving the sun-earth-moon system. Step one creates the various mathematical functions utilized in calculating the orbit at specific intervals. This will be performed with the assistance of the Microsoft Excel spreadsheet program, and the functions will be validated through a singular orbit calculation. Step two incorporates the functions developed in step one into a simple looping computer program. This allows for quicker calculations, self-adaptation, and visual output. During this step, the orbital calculations will be tuned to a finer degree of accuracy. Step three will build on step two by forming a user interface infrastructure around the calculation program. This final program will fulfill the need for an accurate ephemeris generating software which outputs visually and numerically and can be utilized for further astronomical studies.

3.0 Analytical Methodology

3.1 Mathematical Development

Due to the need for an accurate and stable ephemeris generator, it was decided to create a model of the system rather than a simulation. Generally, a simulation is a recreation of spatial interactions, such as velocity, momentum, and acceleration due to gravity, whereas a model makes calculations based upon physical constants. Simulations are not only more complicated, but they are also more likely to become unstable and develop anomalies. A properly executed model, however, will always perform consistently: a major advantage in developing a stable environment for other studies. Although some accuracy is lost, the consistency, dependability, and simplicity of a model more than makes up for this when compared to the inherent instability of a simulation.

To understand the development of this model, one must be familiar with Kepler's laws. The first, and most crucial for this model, states, "The planets move about the sun in ellipses, at one focus of which the sun is situated." The second law states, "The radius vector joining each planet with the sun describes equal areas in equal times." That is, for a given time period the orbiting body "sweeps out" sectors of equal areas. Kepler's third law states, "The cubes of the mean distances of the planets from the sun are equal to the squares of their times of revolution about the sun."[2]

Put simply, the model was developed to fulfill Kepler's first law by projecting the orbits onto a mathematical ellipse. This was accomplished by finding the radius, r, between the sun, f_1, and the body, P, for any given angle, θ, on the ellipse. Figure 3.1 provides a clear visual representation of an elliptical orbit, while also showing the geometric center

of the orbital ellipse, line *AB*, the semimajor axis length, a, and the sun's offset from the geometric center, *c*. Note that in this model, perihelion occurs at θ=0 and aphelion at θ=180, or π radians. The standard ellipse equation must be modified in order to solve for r. By doing so, it is found that

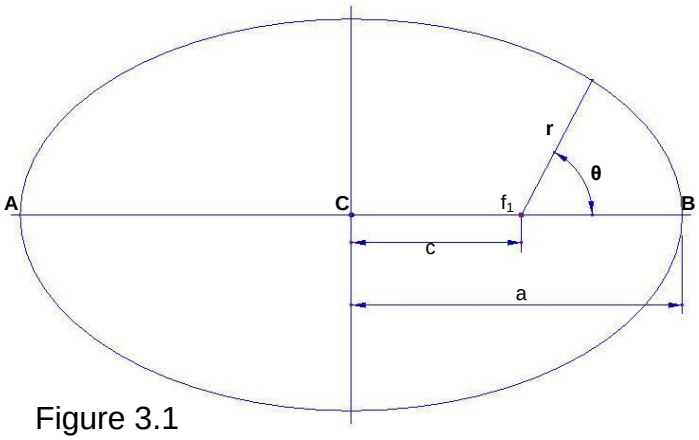


Figure 3.1

$$r = \frac{a(1-e^2)}{1+e\cos\theta}$$ (For a full solution, see Appendix I)

This now allows for the calculation of position at any predefined interval, perhaps one-half of a degree. The smaller the intervals, the better the ellipse's resolution will be and, ultimately, the more accurate the orbit.

At this time it should be noted that, as opposed to popular belief, the earth does not directly orbit the sun. The earth and moon share a common center of mass, which will be referred to as EMCOM from now on, and it is that point which actually orbits the sun. The earth and moon make much smaller orbits around EMCOM. If, as in this project, the goal is to have a heliocentric coordinate system, the positional calculations are performed by splitting the computations into two systems and solving each one independently. Using the radius function described earlier, EMCOM's orbit around the sun can be found. This is already in a heliocentric coordinate system so no further computations are required. The earth-moon system is only slightly more difficult to solve. The desired outcome is for the earth and moon orbits to be defined around EMCOM, that is, with EMCOM as the origin. The difficulty is that current databases describe the moon's orbital parameters in relation to the earth[3]: a geocentric system.

EMCOM's location between the earth and moon is a ratio of each of their singular masses and the total system mass that can be multiplied by the earth-moon radius at any interval to find EMCOM's distance from the chosen body. Thus, the radii between the earth and EMCOM and the moon and EMCOM are found. The angles can remain the same, but it must be noted that the earth and moon will always be 180° opposed to each other. For a better understanding of the above process, see Figure 3.2 or Appendix I. Note that the above processes resulted in the determination of a radius at a specific angle. These are polar coordinates and must be converted to Cartesian coordinates. Recalling Figure 3.1 and that θ=0 occurs at perihelion, EMCOM's x and y positions can be found through the simple trigonometric functions,

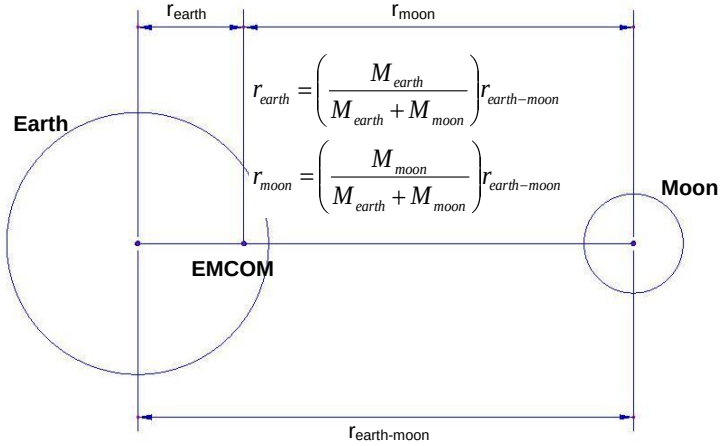


Figure 3.2

$$x = r\cos\theta$$
$$y = r\sin\theta$$

Since the model assumes an isolated, sun-earth-moon system, the angle of the ecliptic can be considered relative and set to zero, so EMCOM's z position also assumes a value of zero. The angle of the moon's orbital plane, however, should be taken into account. That means the previously calculated radii become the absolute radii, or the distance "as the crow flies". With the absolute radius held constant, the true x and y positions projected onto the xy plane change as the angle of the absolute radius is changed. A horizontal radius projection of

the absolute radius onto the xy plane must be found, as in Figure 3.3. This is a cosine function where

$$r_{horizontal} = r_{absolute} \cos\theta_1$$

The x and y positions can then be calculated through the standard method,

$$x = r_{horizontal} \cos\theta_2$$
$$y = r_{horizontal} \sin\theta_2$$

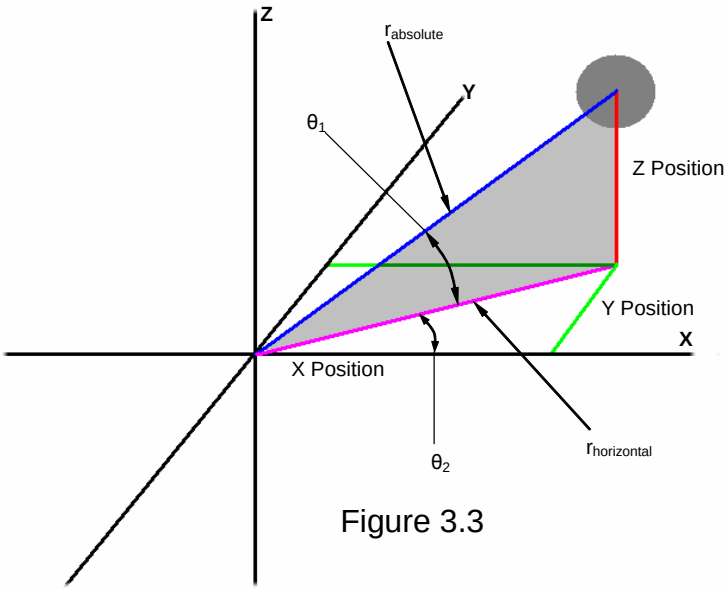


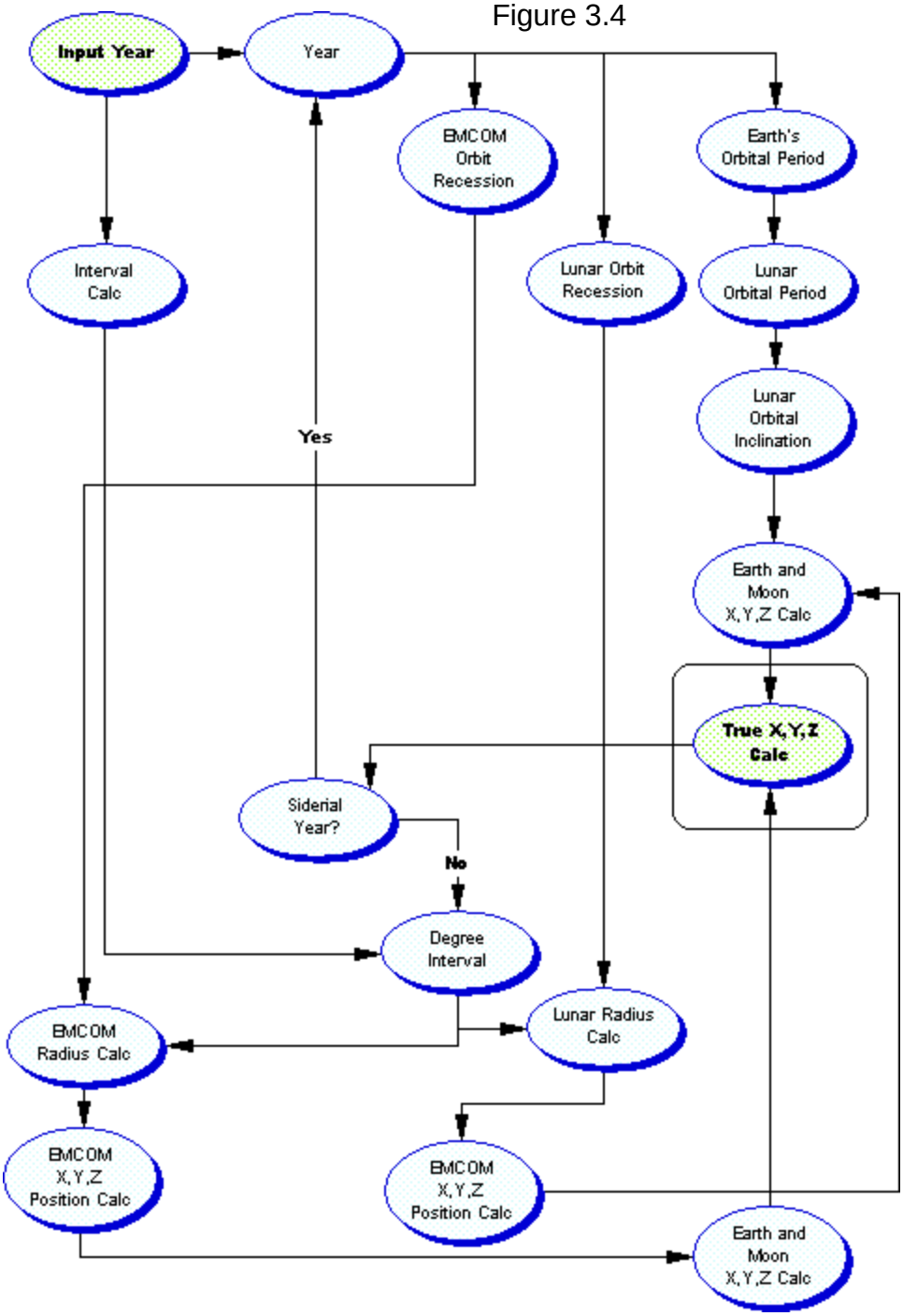
Figure 3.3

These calculations were performed in a spreadsheet program with intervals of less than one degree. It is interesting to note that this solution method is bound by observable, quantifiable data, and that it came about by a unique split-solution method by intervals. This method was completely developed from scratch. For a detailed mathematical solution, see Appendix I.

3.2 Orbital Calculator

As mentioned above, all calculations were first performed in a spreadsheet program. After the calculation of a single EMCOM orbit (one year) it was decided that the project required a more flexible program to perform calculations. A DarkBasic computer program was written which incorporates all of the mathematical models. A flowsheet of the program's operation can be seen in Figure 3.4. During development, there was no intention for this program to have any user input or visual output. However, it was later decided to incorporate visual output to augment numerical output and to allow the user to input a start year for better validation of long-term calculations. The start year should

Figure 3.4



not be confused with a date chosen to begin ephemeris data; rather, it calculates future orbital parameters and applies them to the model. The flexibility of this program exceeds expectations and has allowed for the calculation of various long-term cycles such as precession and orbital recession. Simply put, the program is so flexible that it allows for a level of self-adaptation. Almost all functions are contained in a do loop, so the arrangement of calculations is not as critical as in other programs. Nevertheless, functions requiring input from previous calculations are placed towards

the end of the loop and initial calculations are performed near the beginning. Troubleshooting was done by observing and searching for both visual and numerical inaccuracies. Problems were solved through hand calculations, which were then added to the program, and a recheck performed for similar results between the hand calculations and computer calculations. This program bridges the gap between the mathematical solution methods and the final ephemeris program to be described next.

3.3 User Interface

The user interface combines with the orbital calculator to create a three dimensional program that can be utilized by having basic home computer skills. It allows for a high degree of user control and interaction that the orbital calculator does not and cannot provide. The user interface layout is similar to current Windows type applications, with a menu bar at the top of the screen, a data window with three tabs to select the display of certain information, and a media player-like start/stop control. Along with the ability to fast-forward, rewind, and pause, the user is able to save ephemeris data to printable text files, reload ephemeris data, and reset the program to its default start values. Except for formatting, the numerical output of the user interface is comparable to the orbital calculator. The only numerical input is the ephemeris start date (which can either be the current date and time or a custom date). All mouse action and button selection is also program input and the output is a change in the environment on the screen. For example, if the user selected the “File” button on the menu bar, the output would be the opening of the file menu and the appearance of another selection of buttons. Since the user interface is a program designed to allow the user to interact with the orbital

calculator, proper functioning is validated through trouble free operation of the program, not mathematical comparison.

Specifically designed for this project, the user interface incorporates single line string writing to the software's file management system. This allows for cleaner, more organized files to be written, which in turn is helpful for the user. The program performs numerous repetitive calculations, so it naturally incorporates a repeating loop. A do loop was chosen for its effectiveness and ease of operation. The final performance of the program is reliant on several computer resources. To function correctly, the user interface requires a minimum of 64 MB of RAM, DirectX 7.0, and a Windows operating system.

4.0 Results

4.1 Computer Calculations

With a project such as this, the results of the computer calculations do not necessarily impact the overall final results of the project. Basically, the numerical accuracy of an ephemeris generator does not affect the functioning and stability of the program. This means the orbital calculator's operation directly impacts the final accuracy of the project while the User Interface affects the final function. This two-program solution method greatly reduces the stress on each individual program and results in a much finer piece of software. As such, the results of the computer calculations will be discussed in two parts.

4.1.1 Orbital Calculator

As discussed in a previous section, the orbital calculator incorporates all of the functions originally created in the spreadsheet program. Because of this, the orbital calculator outputs the same data as the excel sheet and hand calculations. Once the long-term changes in orbital parameters were incorporated into the program, the calculator began to continually change and adapt its orbit calculations. Because the calculator was only intended to bridge the gap between calculations and the final user interface, only the current year and positional data is outputted.

Even though the program successfully incorporated the equations from excel, it was not immune to problems and programming flaws. The first major problem occurred while creating a function for calculating and counting the current year. During the first few orbits, all appeared to operate correctly. However, after roughly five orbits, the year counter would begin to count backwards at an exponential rate. Eventually, it was

concluded that the problem was caused by some remaining code from a previous version of the counter creating negative interference for a crucial variable. The problem was solved by deleting the offending lines of code. The second problem had been occurring since the first version of the orbital calculator was written but went unnoticed for a long period of time. It was finally noticed that when EMCOM had made a full orbit ($\theta=360°$), the earth and moon would "jump" to a different position. After much experimentation with different variables, it was found that the glitch was caused by a DarkBasic *wrapvalue* command which reset the interval, θ, to zero when it reached a value of 360. This was causing the earth and moon to reset to their start positions and not continue in their orbit. The *wrapvalue* command was deleted and the problem was solved. The final problem encountered was the most difficult to solve and was known as the "first year" glitch. The glitch became noticeable when the program was set to a start date beyond 3000 A.D. For EMCOM's first orbit, all parameters were calculated for the reference year 2000. In the following orbit, however, the parameters were correctly calculated for the current year, so a large "jump" occurred between the two orbits. Then, during the following orbits, the orbital parameters were recalculated at an exponential rate resulting in an inaccurate and unstable model. It was found that various lines of code in the year counter were adversely affecting the functions calculating the current orbital parameters. A reassignment of variable names finally solved the problem. No other problems have been encountered and the orbital calculator functions extremely well.

4.1.2 User Interface

Once a result has been reached using the interface, there are several functions available to manipulate the data. The user has the option to save the information, to print it as hardcopy, or the possibility to reset the interface. Unlike other ephemeris data, the interface outputs in a simple, easy to understand format. The functions built into DarkBasic allow for a limited file manipulation platform, but by combining those functions with a graphical user interface, the same functionality can be had as major brand software (such as Microsoft office). The most common, and overlooked, problem encountered was the “off by one” error. The “off by one” error occurs when conditional operators are not used correctly. When an internal variable, x, should not be equal to zero, the conditional statement reads,

if x>1 then x=x-1

However, if the statement reads

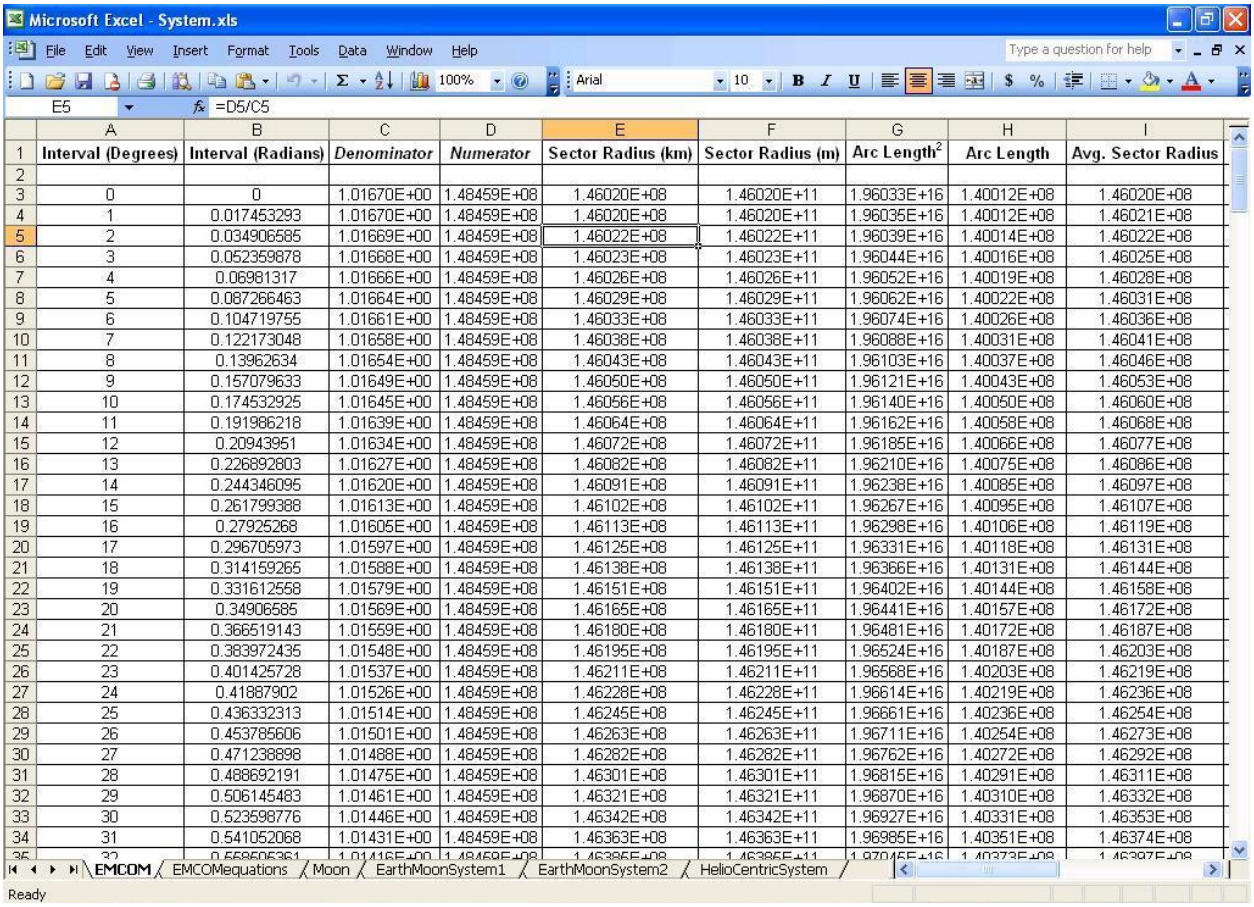
if x≥1 then x=x-1

then eventually x will be set equal to zero. This became a problem when trying to divide by x. Another problem encountered was creating a professional look by applying “sliding parts” – the concept of making images move independently at gradual speeds. This was overcome by applying preset values to a dimension and then using the dimension’s values to move the menu options up and down.

4.2 Graphs, Tables, and Figures

This report incorporates a number of visual aids which were used throughout the project to analyze data and check for correct program function. Other visual aids, such as the program screenshots, are included to provide a better understanding of the visual

quality of the programs. Figure 4.1 is a screenshot of the original excel spreadsheet. Note the one-degree interval calculations. Later calculations, on sheets whose tabs can



	A	B	C	D	E	F	G	H	I
1	Interval (Degrees)	Interval (Radians)	*Denominator*	*Numerator*	Sector Radius (km)	Sector Radius (m)	Arc Length^2	Arc Length	Avg. Sector Radius
2									
3	0	0	1.01670E+00	1.48459E+08	1.46020E+08	1.46020E+11	1.96033E+16	1.40012E+08	1.46020E+08
4	1	0.017453293	1.01670E+00	1.48459E+08	1.46020E+08	1.46020E+11	1.96035E+16	1.40012E+08	1.46021E+08
5	2	0.034906585	1.01669E+00	1.48459E+08	1.46022E+08	1.46022E+11	1.96039E+16	1.40014E+08	1.46022E+08
6	3	0.052359878	1.01668E+00	1.48459E+08	1.46023E+08	1.46023E+11	1.96044E+16	1.40016E+08	1.46025E+08
7	4	0.06981317	1.01666E+00	1.48459E+08	1.46026E+08	1.46026E+11	1.96052E+16	1.40019E+08	1.46028E+08
8	5	0.087266463	1.01664E+00	1.48459E+08	1.46029E+08	1.46029E+11	1.96062E+16	1.40022E+08	1.46031E+08
9	6	0.104719755	1.01661E+00	1.48459E+08	1.46033E+08	1.46033E+11	1.96074E+16	1.40026E+08	1.46036E+08
10	7	0.122173048	1.01658E+00	1.48459E+08	1.46038E+08	1.46038E+11	1.96088E+16	1.40031E+08	1.46041E+08
11	8	0.13962634	1.01654E+00	1.48459E+08	1.46043E+08	1.46043E+11	1.96103E+16	1.40037E+08	1.46046E+08
12	9	0.157079633	1.01649E+00	1.48459E+08	1.46050E+08	1.46050E+11	1.96121E+16	1.40043E+08	1.46053E+08
13	10	0.174532925	1.01645E+00	1.48459E+08	1.46056E+08	1.46056E+11	1.96140E+16	1.40050E+08	1.46060E+08
14	11	0.191986218	1.01639E+00	1.48459E+08	1.46064E+08	1.46064E+11	1.96162E+16	1.40058E+08	1.46068E+08
15	12	0.20943951	1.01634E+00	1.48459E+08	1.46072E+08	1.46072E+11	1.96185E+16	1.40066E+08	1.46077E+08
16	13	0.226892803	1.01627E+00	1.48459E+08	1.46082E+08	1.46082E+11	1.96210E+16	1.40075E+08	1.46086E+08
17	14	0.244346095	1.01620E+00	1.48459E+08	1.46091E+08	1.46091E+11	1.96238E+16	1.40085E+08	1.46097E+08
18	15	0.261799388	1.01613E+00	1.48459E+08	1.46102E+08	1.46102E+11	1.96267E+16	1.40095E+08	1.46107E+08
19	16	0.27925268	1.01605E+00	1.48459E+08	1.46113E+08	1.46113E+11	1.96298E+16	1.40106E+08	1.46119E+08
20	17	0.296705973	1.01597E+00	1.48459E+08	1.46125E+08	1.46125E+11	1.96331E+16	1.40118E+08	1.46131E+08
21	18	0.314159265	1.01588E+00	1.48459E+08	1.46138E+08	1.46138E+11	1.96366E+16	1.40131E+08	1.46144E+08
22	19	0.331612558	1.01579E+00	1.48459E+08	1.46151E+08	1.46151E+11	1.96402E+16	1.40144E+08	1.46158E+08
23	20	0.34906585	1.01569E+00	1.48459E+08	1.46165E+08	1.46165E+11	1.96441E+16	1.40157E+08	1.46172E+08
24	21	0.366519143	1.01559E+00	1.48459E+08	1.46180E+08	1.46180E+11	1.96481E+16	1.40172E+08	1.46187E+08
25	22	0.383972435	1.01548E+00	1.48459E+08	1.46195E+08	1.46195E+11	1.96524E+16	1.40187E+08	1.46203E+08
26	23	0.401425728	1.01537E+00	1.48459E+08	1.46211E+08	1.46211E+11	1.96568E+16	1.40203E+08	1.46219E+08
27	24	0.41887902	1.01526E+00	1.48459E+08	1.46228E+08	1.46228E+11	1.96614E+16	1.40219E+08	1.46236E+08
28	25	0.436332313	1.01514E+00	1.48459E+08	1.46245E+08	1.46245E+11	1.96661E+16	1.40236E+08	1.46254E+08
29	26	0.453785606	1.01501E+00	1.48459E+08	1.46263E+08	1.46263E+11	1.96711E+16	1.40254E+08	1.46273E+08
30	27	0.471238898	1.01488E+00	1.48459E+08	1.46282E+08	1.46282E+11	1.96762E+16	1.40272E+08	1.46292E+08
31	28	0.488692191	1.01475E+00	1.48459E+08	1.46301E+08	1.46301E+11	1.96815E+16	1.40291E+08	1.46311E+08
32	29	0.506145483	1.01461E+00	1.48459E+08	1.46321E+08	1.46321E+11	1.96870E+16	1.40310E+08	1.46332E+08
33	30	0.523598776	1.01446E+00	1.48459E+08	1.46342E+08	1.46342E+11	1.96927E+16	1.40331E+08	1.46353E+08
34	31	0.541052068	1.01431E+00	1.48459E+08	1.46363E+08	1.46363E+11	1.96985E+16	1.40351E+08	1.46374E+08

Figure 4.1

be seen on the bottom left portion of the image, were performed using smaller intervals. To check that data and the correct functioning of the radius equation, a graph was made of all calculated radii between zero and 360 degrees. Since the path described by an ellipse is basically a modified circle, the graph was expected to output a modified sine or cosine curve. Figure 4.2 shows the results and validates correct functioning of the radius equation. To double-check these results, the spreadsheet data was exported to a Microsoft Notepad document. That document was then incorporated into a two dimensional GnuPlot graph. As can be seen in Figure 4.3, the data correctly defines an

elliptical orbit. Once the conversion between polar and Cartesian coordinates was made, the x and y positions were graphed over an interval between 0 and 360 degrees. In Figure 4.4, note how the x and y locations correctly follow the $x = r\cos\theta$ and $y = r\sin\theta$ equations. Upon completion of the orbital calculator, positional data for one year was exported to Notepad and incorporated into a series of two and three-dimensional GnuPlot diagrams. Figures 4.5 and 4.6 show the results. Observe how the moon performs a series of loops about EMCOM which look like flower petals. This is not how the moon truly orbits EMCOM, and the error is due to different scale factors between the earth-moon orbit and the EMCOM-sun orbit. Figures 4.7 and 4.8 show the results of

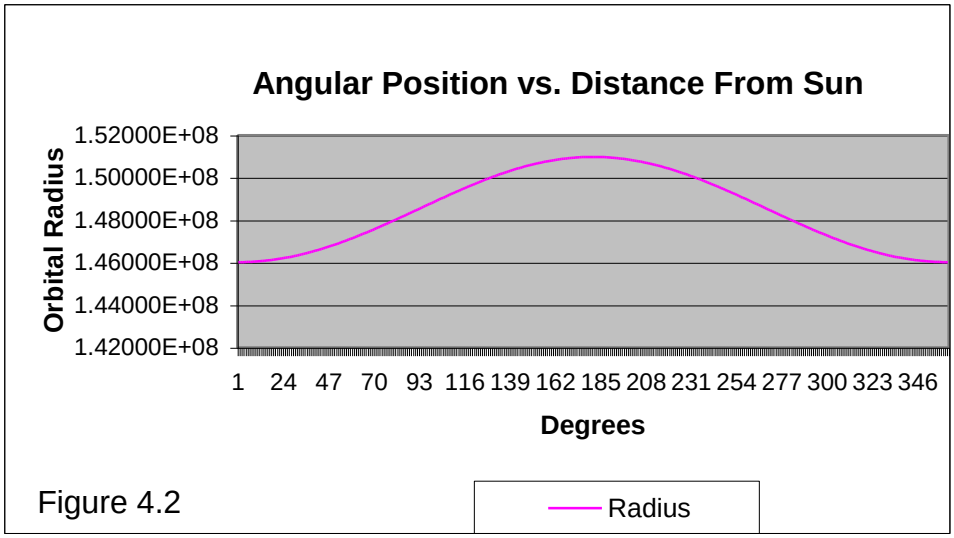


Figure 4.2

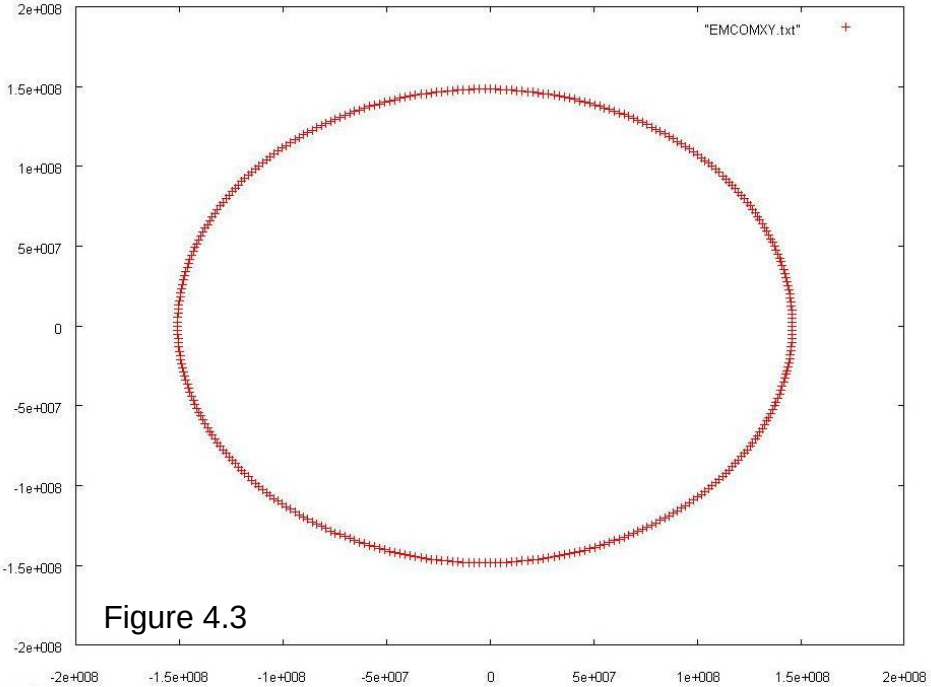
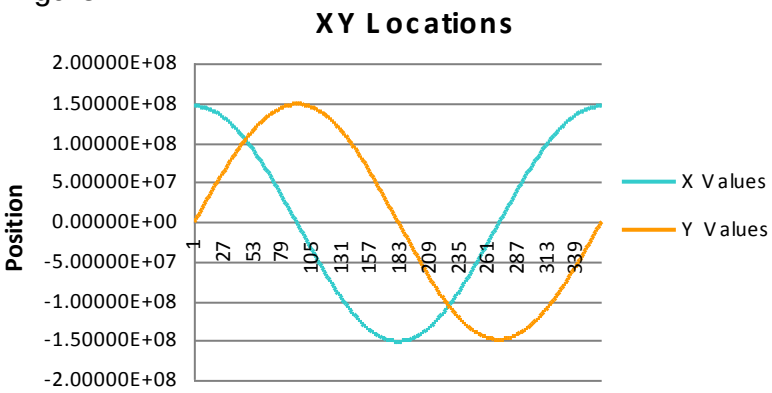


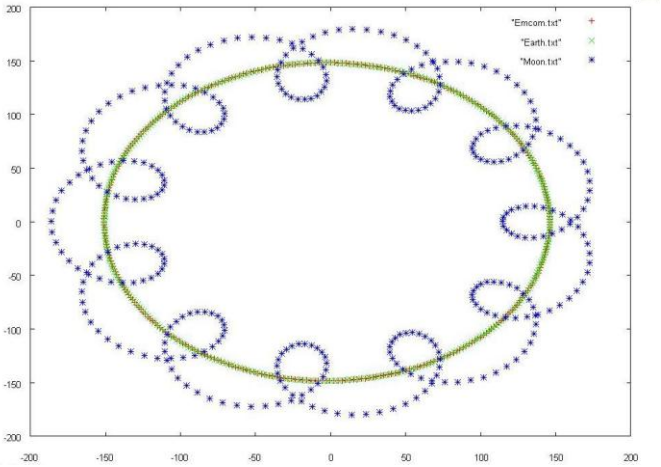
Figure 4.3

Figure 4.4



a correctly scaled orbit. A screenshot of the orbital calculator, Figure 4.9, was taken shortly after visual output was added to the program. Compared to the User Interface in Figures 4.10, 4.11, and 4.12, the orbital calculator has very simplistic visual output. The screenshots show the progression of the User Interface through its various forms of development. The images are in chronological order, with the last image showing its final form.

Figure 4.5



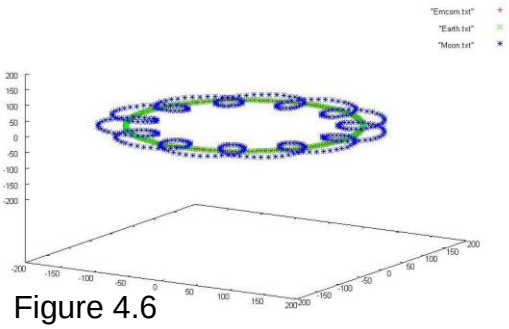


Figure 4.6

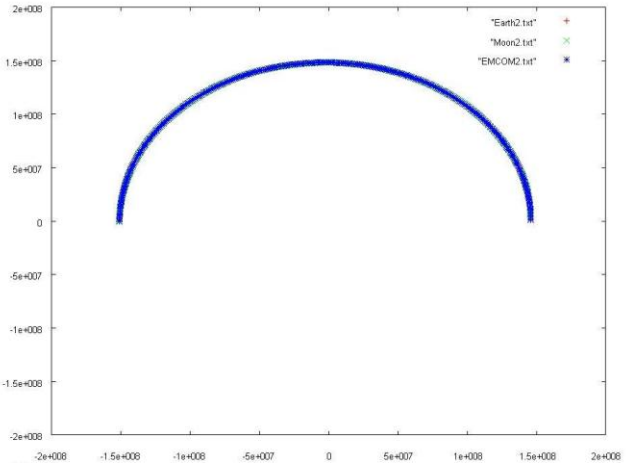


Figure 4.7

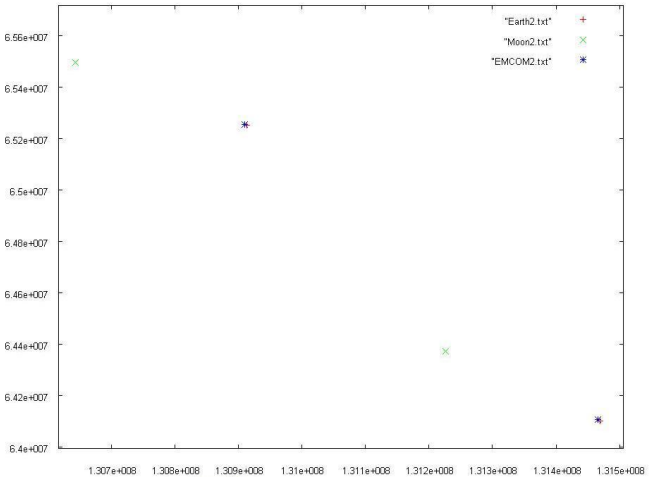


Figure 4.8

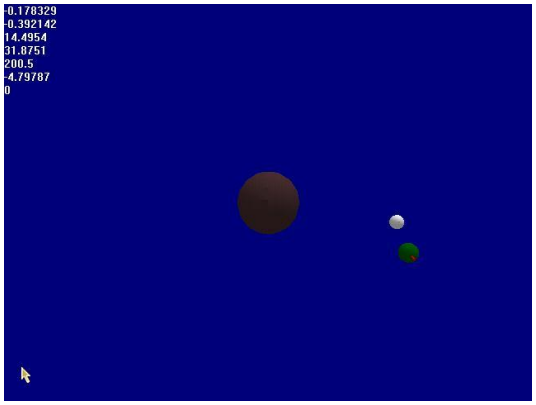


Figure 4.9

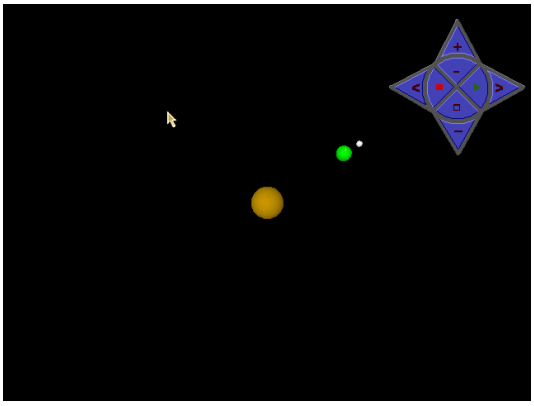


Figure 4.10

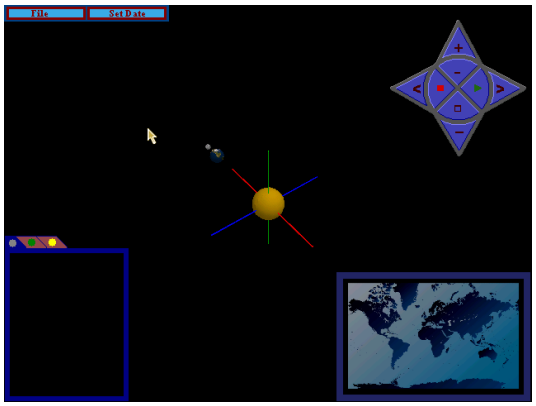


Figure 4.11

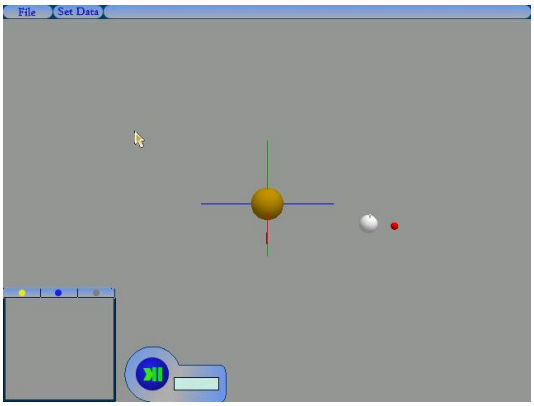


Figure 4.12

4.3 Other Data

The following data was taken from NASA's Horizons Web Interface ephemeris generator[4]. It is shown here as an example of the most common ephemeris format: an observer-type ephemeris given in spherical coordinates. Note how this would appear confusing, or even unintelligible, to anyone except the knowledgeable professional.

****

Revised: Mar 11, 1998 Moon / (Earth) 301

PHYSICAL PROPERTIES:

Radius, km = 1737.53+-0.03 Mass, 10^20 kg = 734.9

Density, gm cm^-3 = 3.3437 Geometric albedo = 0.12

V(1,0) = +0.21 GM, km^3/s^2 = 4902.798+-.005

Earth/Moon mass ratio = 81.300587 Surface gravity = 1.62 m s^-2

Nearside crust. thick.= 58+-8 km Farside crust. thick. = ~80 - 90 km

Heat flow, Apollo 15 = 3.1+-.6 mW/m^2 Heat flow, Apollo 17 = 2.2+-.5 mW/m^2

Mean crustal density = 2.97+-.07g/cm^3 k2 = 0.0302+-.0012

Induced magnetic mom. = 4.23x10^22Gcm^3 Magnetometer moment = 435+-15

DYNAMICAL CHARACTERISTICS:

Mean angular diameter = 31'05.2" Orbit period = 27.321582 d

Obliquity to orbit = 6.67 deg Eccentricity = 0.05490

Semi-major axis, a = 384400 km Inclination = 5.145 deg

Mean motion, rad/s = 2.6616995x10^-6 Nodal period = 6798.38 d

Apsidal period = 3231.50 d Mom. of inertia C/MR^2= 0.3935+-.0011

beta (C-A/B), x10^-4 = 6.31(72+-15) gamma (B-A/C), x10^-4 = 2.278(8+-2)

```
*******************************************************************************
Ephemeris / WWW_USER Sat Mar 28 18:47:32 2009  Pasadena, USA     / Horizons
*******************************************************************************
Target body name: Moon (301)                {source: DE405}
Center body name: Earth (399)               {source: DE405}
Center-site name: GEOCENTRIC
*******************************************************************************
Start time      : A.D. 2000-Jan-01 00:00:00.0000 UT
Stop  time      : A.D. 2000-Jan-02 00:00:00.0000 UT
Step-size       : 120 minutes
*******************************************************************************
Target pole/equ : IAU_MOON                  {East-longitude +}
Target radii    : 1737.4 x 1737.4 x 1737.4 km     {Equator, meridian, pole}
Center geodetic : .000000000,.000000000,.00000000 {E-lon(deg),Lat(deg),Alt(km)}
Center cylindric: .000000000,.000000000,.00000000 {E-lon(deg),Dxy(km),Dz(km)}
Center pole/equ : High-precision EOP model       {East-longitude +}
Center radii    : 6378.1 x 6378.1 x 6356.8 km     {Equator, meridian, pole}
Target primary  : Earth                     {source: DE405+DE406}
Interfering body: MOON (Req= 1737.400) km        {source: DE405}
Deflecting body : Sun, EARTH                {source: DE405}
Deflecting GMs  : 1.3271E+11, 3.9860E+05 km^3/s^2
Atmos refraction: NO (AIRLESS)
RA format       : HMS
```

Time format : CAL

EOP file : eop.090327.p090618

EOP coverage : DATA-BASED 1962-JAN-20 TO 2009-MAR-27. PREDICTS-> 2009-JUN-17

Units conversion: 1 AU= 149597870.691 km, c= 299792.458 km/s, 1 day= 86400.0 s

Table cut-offs 1: Elevation (-90.0deg=NO),Airmass (>38.000=NO), Daylight (NO)

Table cut-offs 2: Solar Elongation (0.0,180.0=NO)

****

Date__(UT)__HR:MN R.A._(ICRF/J2000.0)_DEC Ang-diam r rdot delta deldot S-O-T /r S-T-O

****

$$SOE

2000-Jan-01 00:00 14 26 42.18 -08 59 49.3 1787.817 0.982105341127 -0.8286693 .002679833220549 0.0381841 62.6897 /L 117.1713

2000-Jan-01 02:00 14 30 33.05 -09 19 20.6 1786.608 0.982065643289 -0.8209402 .002681646653398 0.0371797 61.7713 /L 118.0908

2000-Jan-01 04:00 14 34 24.08 -09 38 41.3 1785.433 0.982026322196 -0.8130141 .002683411743824 0.0361751 60.8540 /L 119.0091

2000-Jan-01 06:00 14 38 15.28 -09 57 51.0 1784.292 0.981987387272 -0.8048933 .002685128502044 0.0351711 59.9379 /L 119.9264

2000-Jan-01 08:00 14 42 06.68 -10 16 49.6 1783.184 0.981948847830 -0.7965803 .002686796962657 0.0341680 59.0230 /L 120.8425

2000-Jan-01 10:00 14 45 58.29 -10 35 36.7 1782.109 0.981910713064 -0.7880773
.002688417183958 0.0331663 58.1090 /L 121.7577
2000-Jan-01 12:00 14 49 50.14 -10 54 12.2 1781.067 0.981872992060 -0.7793869
.002689989247251 0.0321666 57.1961 /L 122.6718
2000-Jan-01 14:00 14 53 42.24 -11 12 35.6 1780.059 0.981835693784 -0.7705112
.002691513256148 0.0311693 56.2842 /L 123.5850
2000-Jan-01 16:00 14 57 34.61 -11 30 46.9 1779.083 0.981798827093 -0.7614528
.002692989335875 0.0301748 55.3732 /L 124.4973
2000-Jan-01 18:00 15 01 27.27 -11 48 45.6 1778.140 0.981762400726 -0.7522139
.002694417632561 0.0291835 54.4632 /L 125.4088
2000-Jan-01 20:00 15 05 20.24 -12 06 31.6 1777.229 0.981726423310 -0.7427970
.002695798312537 0.0281959 53.5540 /L 126.3194
2000-Jan-01 22:00 15 09 13.52 -12 24 04.6 1776.351 0.981690903356 -0.7332045
.002697131561616 0.0272124 52.6456 /L 127.2292
2000-Jan-02 00:00 15 13 07.14 -12 41 24.3 1775.504 0.981655849261 -0.7234387
.002698417584386 0.0262333 51.7380 /L 128.1382
$$EOE
****

Column meaning:

TIME

Prior to 1962, times are UT1. Dates thereafter are UTC. Any 'b' symbol in the 1st-column denotes a B.C. date. First-column blank (" ") denotes an A.D. date. Calendar dates prior to 1582-Oct-15 are in the Julian calendar system.

Later calendar dates are in the Gregorian system.

The uniform Coordinate Time scale is used internally. Conversion between CT and the selected non-uniform UT output scale has not been determined for UTC times after the next July or January 1st. The last known leap-second is used over any future interval.

NOTE: "n.a." in output means quantity "not available" at the print-time.

R.A._(ICRF/J2000.0)_DEC =

J2000.0 astrometric right ascension and declination of target center. Corrected for light-time. Units: HMS (HH MM SS.ff) and DMS (DD MM SS.f)

Ang-diam =

The equatorial angular width of the target body full disk, if it were fully visible to the observer. Units: ARCSECONDS

r rdot =

Heliocentric range ("r", light-time corrected) and range-rate ("rdot") of the target center at the instant light seen by the observer at print-time would have left the target center (print-time minus down-leg light-time). The Sun-to-target distance traveled by a ray of light emanating from the center of the Sun that reaches the target center point at some instant and is recordable by the observer one down-leg light-time later at print-time. Units: AU and KM/S

delta deldot =

Range ("delta") and range-rate ("delta-dot") of target center with respect to the observer at the instant light seen by the observer at print-time would

have left the target center (print-time minus down-leg light-time); the distance traveled by a light ray emanating from the center of the target and recorded by the observer at print-time. "deldot" is a projection of the velocity vector along this ray, the light-time-corrected line-of-sight from the coordinate center, and indicates relative motion. A positive "deldot" means the target center is moving away from the observer (coordinate center). A negative "deldot" means the target center is moving toward the observer.

Units: AU and KM/S

S-O-T /r =

Sun-Observer-Target angle; target's apparent solar elongation seen from observer location at print-time. If negative, the target center is behind the Sun. Angular units: DEGREES.

The '/r' column is a Sun-relative code, output for observing sites with defined rotation models only.

/T indicates target trails Sun (evening sky)

/L indicates target leads Sun (morning sky)

NOTE: The S-O-T solar elongation angle is the total separation in any direction. It does not indicate the angle of Sun leading or trailing.

S-T-O =

Sun-Target-Observer (~ PHASE ANGLE) angle: the vertex angle at target center formed by a vector to the apparent center of the Sun and a vector intersecting the observer at print-time. This measurable angle is within 20 arcseconds (0.006 deg) of the reduced PHASE ANGLE at observer's location at print time.

The difference is due to down-leg stellar aberration affecting measured target position but not apparent solar illumination direction. When computing phase, Horizons uses the true phase angle, not S-T-O, but the resulting difference in illuminated fraction is less than 0.001%.

Units: DEGREES

Computations by ...

Solar System Dynamics Group, Horizons On-Line Ephemeris System

4800 Oak Grove Drive, Jet Propulsion Laboratory

Pasadena, CA 91109 USA

Information: http://ssd.jpl.nasa.gov/

Connect : telnet://ssd.jpl.nasa.gov:6775 (via browser)

telnet ssd.jpl.nasa.gov 6775 (via command-line)

Author : Jon.Giorgini@jpl.nasa.gov

****

5.0 Conclusion

5.1 Mathematical Models

There is not a clear cut way to model the Sun-Earth-Moon system, or any celestial motion for that matter, and many methods have been implemented over the years to track the movements of bodies in space. Two different methods were experimented with prior to reaching the desired mathematical precision and accuracy. First a vector model was created, trying to recreate the system solely by calculating acceleration due to gravity for the X, Y, Z components of the earth and moon. The results of this program were sporadic at best, so this method was discontinued. The next method that was tried was a location calculation algorithm that based positions upon previous locations by calculating change in location, then adding to the current location. At first this method seemed promising; however, when running the program through a long time period the effects of Euler's Theorem were increasingly noticeable so this method was deemed inadequate. The final method that was explored, and later utilized, was a location calculation based upon an angle from a reference axis. This method, which is fully described in Appendix I, was specially developed for this project. By using the Riemann Sum strategy for approximation, this interval method allowed for great accuracy and minimal error in the calculations, rendering a more than satisfactory mathematical model that could easily be integrated into the computer program.

5.2 Program

With the goal to create user-friendly ephemeris software, the user interface performed as expected and is a success. It is easy to use and provides functionality that makes it a good choice for amateurs. The intuitive functions included in the interface make it so

that anyone can understand and use it. After completing the user interface, it was restructured to be compatible with a wider variety of computers. This was accomplished by implementing new image mapping techniques which use less RAM, and by making use of DirectX's DirectDraw to provide a widely accepted program.

5.3 Results

As was proposed, this project successfully functions as a stable model of the Sun-Earth-Moon system and returns ephemeris data in both a numerical and visual manner. The model can also be used for further Sun-Earth-Moon investigations, as it is a continually adapting program. The three-step method that was described in the Project Proposal proved to be a successful way to complete the project. By utilizing the appropriate mathematical functions and calculating them in Excel, a singular orbit was described. Incorporating the functions and changes in orbits as constants into a looping program, referred to as the Orbital Calculator, a tremendous amount of calculations were made. When compared to NASA's ephemerides, the calculations were found to be correct. The user interface was then constructed to incorporate the raw computations of the Orbital Calculator into a user-friendly environment that could be utilized by both amateurs and professionals.

5.4 Word Processor Program

For the writing of this technical report, Microsoft Word was used because of its ability to format text, enter mathematical equations, and import diagrams to provide visuals that corroborate with the written explanations.

5.5 Recommendations

A useful addition to the program would be to have it calculate and return physical attributes of the system such as gravitational attraction, angular momentum, tangential velocities, and other similar values. Although items such as these do not affect the performance of the program, they could be useful in obtaining a better physical understanding of the Sun-Earth-Moon system.

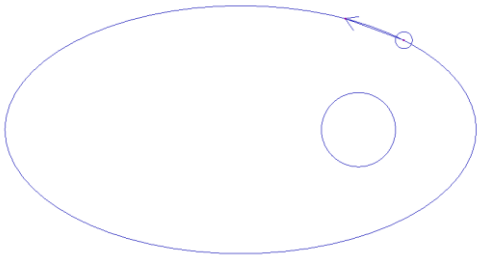
References

1. “Spherical Coordinates.” AbsoluteAstronomy.com. 27 Mar. 2009 http://absoluteastronomy.com/topics/Spherical_coordinate_system.
2. Weast, Robert C., ed. CRC Handbook of Chemistry and Physics. 58th ed. Cleveland: CRC P, 1977.
3. "Moon Fact Sheet." Welcome to the NSSDC! 19 Dec. 2008 http://nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html.
4. "HORIZONS System." JPL Solar System Dynamics. 19 Dec. 2008 http://ssd.jpl.nasa.gov/?horizons.
5. Weisstein, Eric W. “Ellipse.” From MathWorld—A Wolfram Web Resource. http://mathworld.wolfram.com/Ellipse .html
6. "Calculating Precise Long Term Stellar Motions | The Astronomy Nexus." The Astronomy Nexus | science, art, and advice. 19 Dec. 2008 http://www.astronexus.com/node/40.
7. "Earth Fact Sheet." Welcome to the NSSDC! 19 Dec. 2008 http://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html.

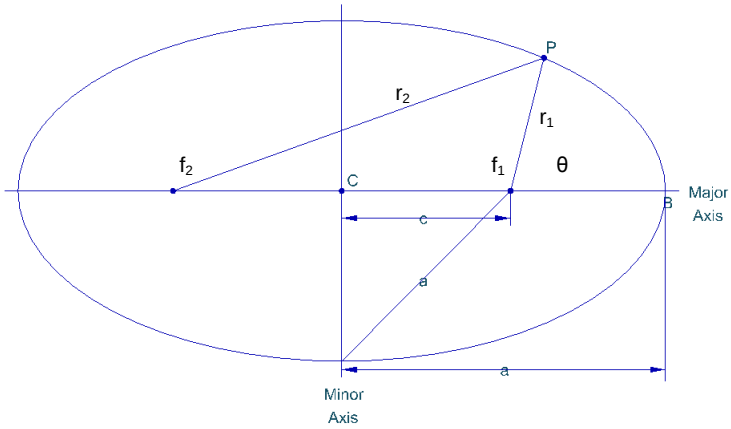
Appendix I Equations and Mathematical Procedures

A1.1 Mathematical Bases

In order to correctly define the motion of the earth and moon about the sun, one must first have a strong grasp of the interactions involved and how to best describe them. According to Kepler's first law, all orbiting bodies follow an elliptical path:

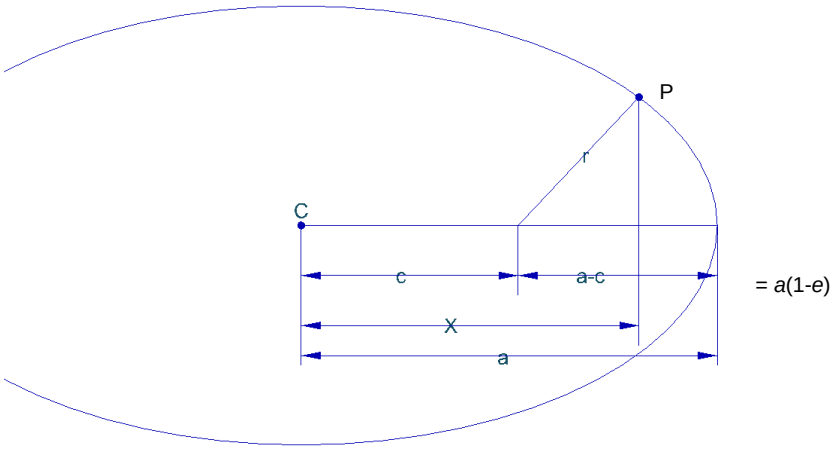


This path can be defined by:



Where *P* is a point on the ellipse, *a* is the semimajor axis length, *c* is the distance between the geometric center, *C*, and focus point, f_1. F_1 and f_2 are focus points, while r_1 and r_2 are the respective radii between said points. Theta is the angle formed between

line *CB* and r_1. In actuality, the earth and moon orbit around their common center of mass, EMCOM, and it is that point which follows an elliptical orbit about the sun. The sun is the focus point in EMCOM's orbit, so f_1 is redefined as the center of the sun and r_1 becomes the radius between the sun and EMCOM. It is that radius which must be found for all points, *P*, about the sun:



Given the standard equation of an ellipse,

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$

x and *y* can be redefined as:

$$x = c + r\cos\theta$$
$$y = r\sin\theta$$

and the equation becomes

$$\frac{(c+r\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2}=1$$
$$=\frac{c^2+2cr\cos\theta+r^2\cos^w\theta}{a^2}+\frac{r^2\sin^2\theta}{b^2}$$

Clearing the denominators gives,

$$b^2(c^2 + 2cr\cos\theta + r^2\cos^2\theta) + a^2r^2\sin^2\theta = a^2b^2$$

Substituting $\sin^2\theta = 1 - \cos^2\theta$ and redefining *b* and *c* in terms of *a* and *e*,

$$a^2(1-e^2)a^2e^2 + 2aea^2(1-e^2)r\cos\theta + a^2(1-e^2)r^2\cos^2\theta + a^2r^2\cos^2\theta = a^2[a^2(1-e^2)]$$

Dividing by $-a^2$ and simplifying,

$$-r^2 + [r\cos\theta - a(1-e^2)]^2 = 0$$

Solving for *r* gives[5],

$$r = \pm[r\cos\theta - a(1-e^2)]$$
$$= a(1-e^2) - er\cos\theta$$
$$r(1+e\cos\theta) = a(1-e^2)$$

$$r = \frac{a(1-e^2)}{1+e\cos\theta}$$

Now knowing r, two simple checks for accuracy can be had. Both require two radius lengths, r_1 and r_2, at two thetas, θ_1 and θ_2, respectively. Ideally, the difference between θ_1 and θ_2 should be relatively small (less than one degree) for best accuracy. Assuming that the difference is small enough so that the curvature of the ellipse becomes a straight line, the length of the arc between r_1 and r_2 can be had through:

$$L_{arc}^{\ 2} = r_1^{\ 2} + r_2^{\ 2} - 2r_1r_2\cos(\theta_2 - \theta_1)$$
$$L_{arc} = \sqrt{r_1^{\ 2} + r_2^{\ 2} - 2r_1r_2\cos(\theta_2 - \theta_1)}$$

The accuracy can be checked through this method by taking the above formula and calculating the sum of all intervals between 0 and 360 degrees,

$$C = \sum_0^{360}\sqrt{r_1^{\ 2} + r_2^{\ 2} - 2r_1r_2\cos(\theta_2 - \theta_1)}$$

Then, compare against any of the commonly used approximation formulas. The following method, however, is more efficient than the previous. As stated above, this method also begins with r_1 and r_2 at two thetas. This time, an approximate area of the sector will be found through:

$$r_{sector} = \frac{r_1 r_2}{2}$$

Where r_{sector} is the average radius of the calculated sector. The sector's area can be had by,

$$A_{\text{sector}} = (\frac{\theta}{360})[\pi(r_{\text{sector}}^2)]$$

And the ellipse's total area can be found through:

$$A_{ellipse} = \sum_0^{360}[(\frac{\theta}{360})(\pi r_{\text{sector}_i}^2)]$$

This can then be compared to the simple area function,

$$A = \pi ab$$

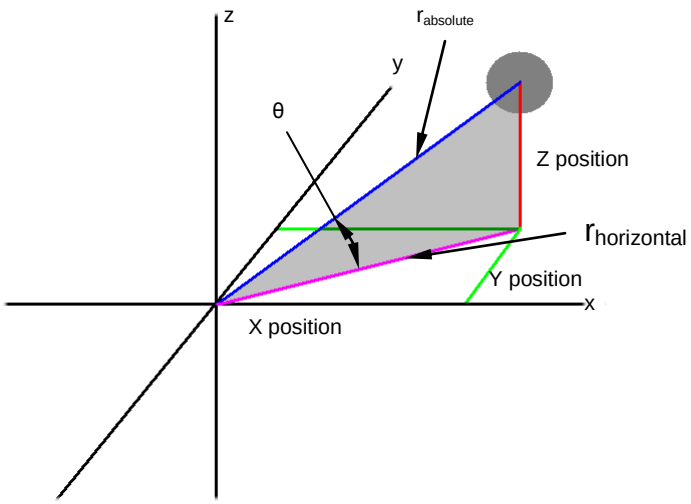
where *a* is the length of the semimajor axis and *b* is the length of the semiminor axis. The same functions used to describe EMCOM's orbit around the sun can be applied to calculate the moon's distance from the earth. Because the earth and moon orbit about EMCOM, EMCOM's position must be calculated at each interval. To do so EMCOM's location must be calculated as a fraction of the total distance between the earth and moon:

$$r_{emcom} = \left(\frac{M_{moon}}{M_{earth} + M_{moon}} \right) r_{earth/moon}$$

Now earth's and moon's radius can be calculated with EMCOM as the center point,

$$r_{moon} = r_{earth-moon} - r_{emcom}$$
$$r_{earth} = r_{emcom}$$

Note that the calculated radii of the earth and moon are the absolute radii; therefore, the true x, y, and z positions must still be found because the inclination of the moon's orbit causes the x and y positions to be somewhat less than the absolute radius. To better understand this, consider a right triangle: according to the Pythagorean Theorem, the hypotenuse will *always* be greater than any one of the triangle's legs:



Having found the absolute radius, r_{earth} or r_{moon}, and knowing the inclination of the moon's orbital plane (this cycles between 5.1 degrees and -5.1 degrees with each orbit, so the angle follows a modified sine function in relation to the interval location), the horizontal component of the x and y locations, $r_{horizontal}$, can be found through:

$$r_{horizontal} = r_{absolute} \cos\theta$$

Then it is a simple matter of trigonometry to solve for the x and y positions,

$$x = \pm r_{horizontal} \cos\theta$$
$$y = \pm r_{horizontal} \sin\theta$$
$$z = \pm\sqrt{r_{absolute}^2 - r_{horizontal}^2} = \pm r_{absolute} \sin\theta$$

Because it is assumed EMCOM orbits in a flat plane, the above calculations are not needed to determine EMCOM's position. Also note that all methods are converting polar coordinates, a length and angle, into Cartesian coordinates. The final step in determining the bodies' positions is to corroborate all points in a heliocentric coordinate system. This is relatively simple, as EMCOM has already been calculated in a heliocentric environment. Complications are further reduced because the earth-moon system was calculated with EMCOM as the origin. Thus, the true positions can be determined through:

$$Earth_x = x_{emcom} + x_{earth}$$
$$Earth_y = y_{emcom} + y_{earth} \qquad Moon_z = z_{emcom} + z_{moon}$$
$$Earth_z = z_{emcom} + z_{earth} \qquad EMCOM_x = x_{emcom}$$
$$Moon_x = x_{emcom} + x_{moon} \qquad EMCOM_y = y_{emcom}$$
$$Moon_y = y_{emcom} + y_{moon} \qquad EMCOM_z = 0$$

Annual changes in long term cycles such as recession and precession can also be calculated. Since the recession rate of the earth-moon system is generally given in terms of the moon, a ratio must be found between the earth and moon's masses and multiplied by the moon's recession rate:

$$recession_{earth} = rate\left(\frac{1}{M_{earth} / M_{moon}}\right)$$

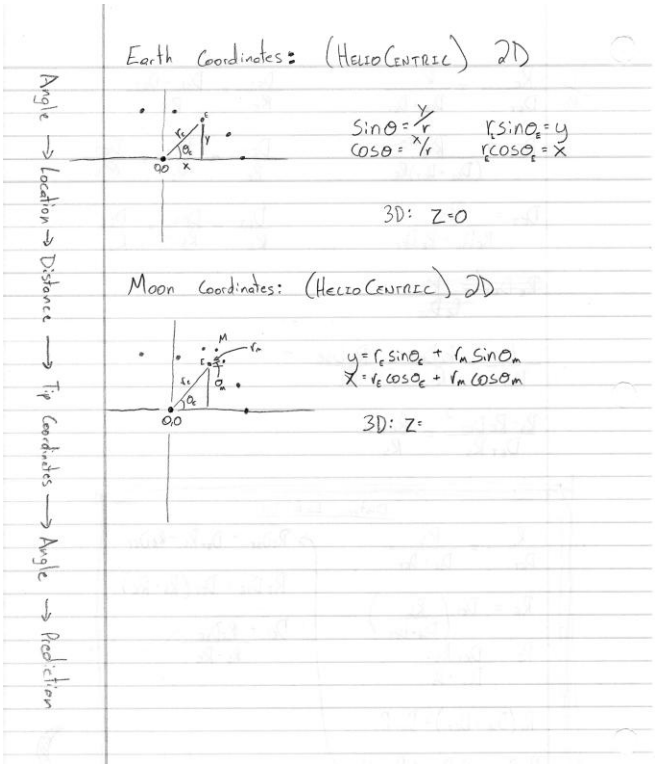
Earth's precession period is roughly 26,000 years, so an annual value can be found through,

$$precession_{annual} = \left(\frac{1}{period}\right)360°$$

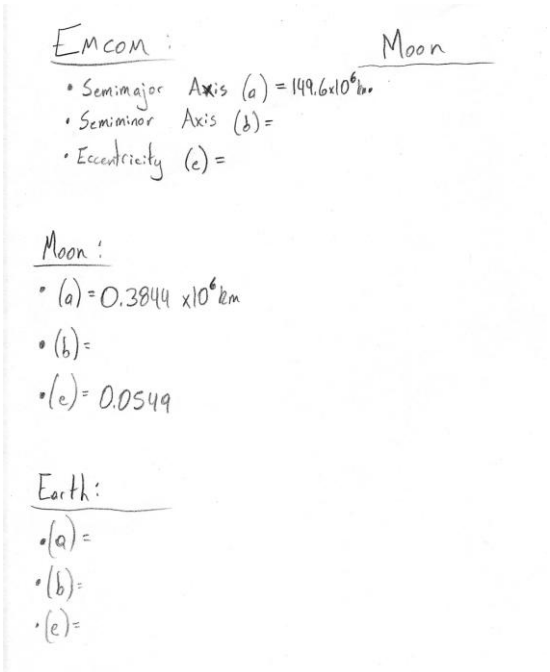
Appendix II Hand Calculations

A2.1 Work Examples

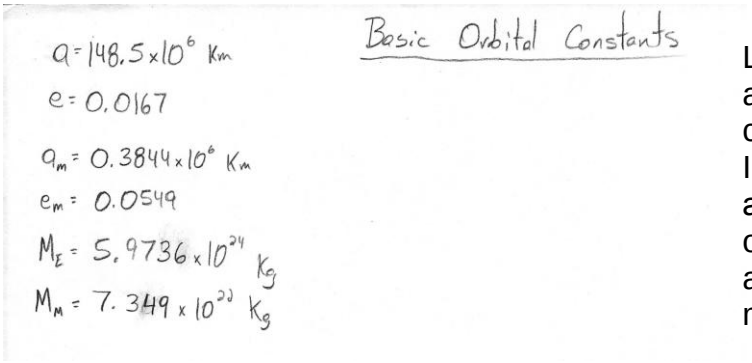
Most hand calculations were performed on media not easily digitized or in a manner that would be of little use to the reader. As such, only a few examples of this were are presented here. A short description of the work accompanies each image.



This sheet shows early experimentation with the two-system method of solution. Although the position calculations are not quite sound, the fundamental idea behind the work is correct.

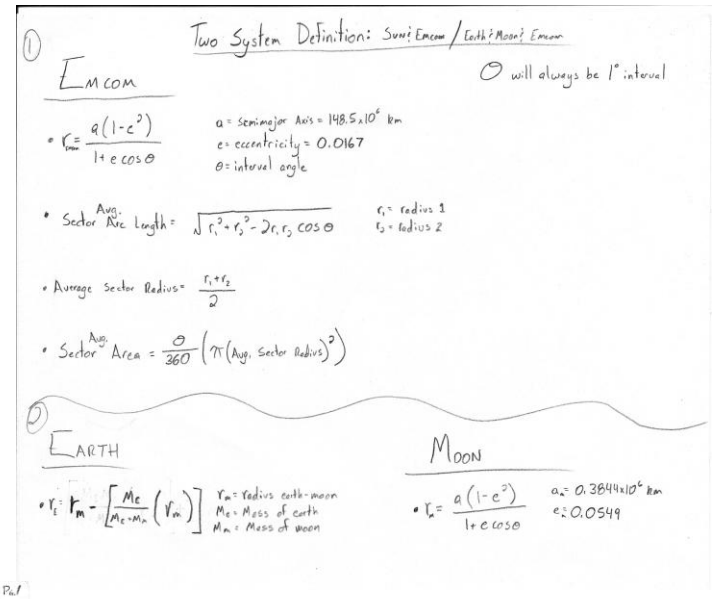


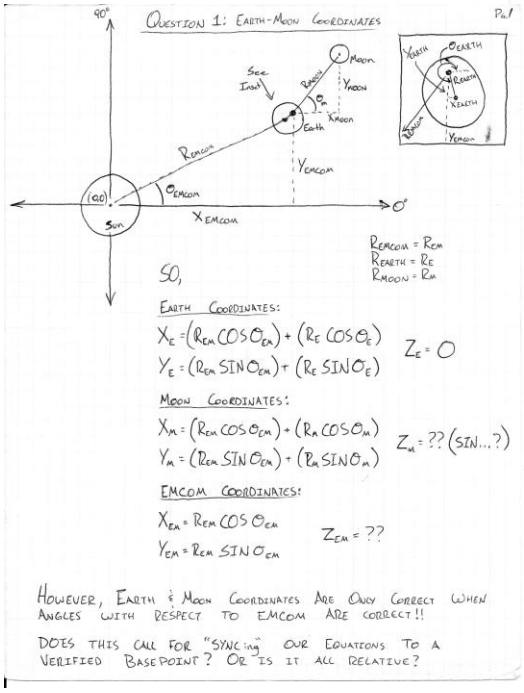
After conceiving the two-system solution method, orbital parameters for the earth, moon, and EMCOM were gathered. This is an example of an unfinished sheet documenting those parameters.



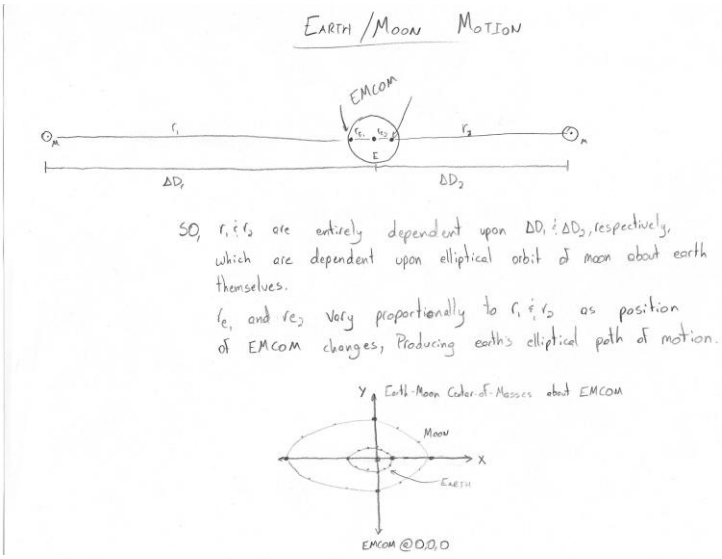
Like the sheet above, this is another with orbital constant/parameter details. Included in this sheet is the average radius of EMCOM's orbit, its orbital eccentricity, and the mass of the earth and moon.

This sheet correctly defined the two systems. The description of one-degree intervals indicates that this is still a relatively early piece of work. Also, note how the position of EMCOM within the earth-moon system has not yet been described.



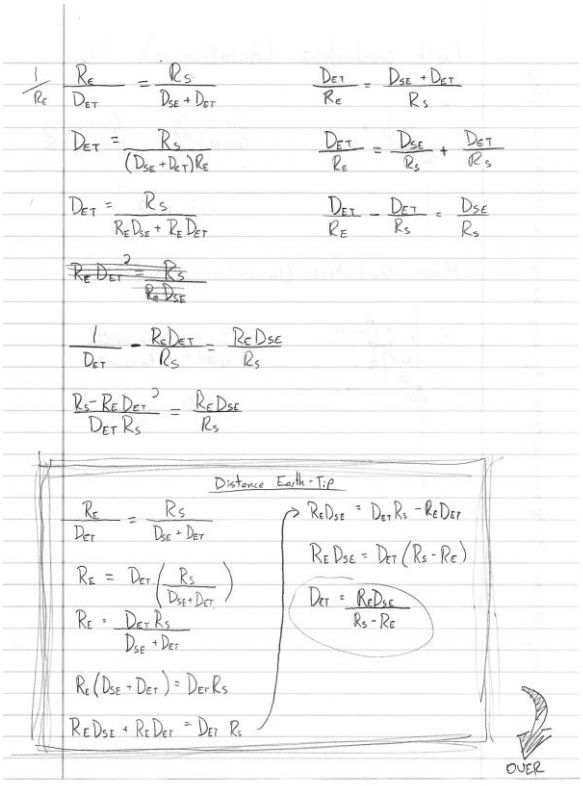


This sheet was composed as a question for the team mentor. After correctly defining the x and y components of the two-system solution, uncertainty still existed around the z component and how the system was synced with the sun-EMCOM system.



This sheet describes the logic behind the solution for EMCOM's location within the earth-moon system. Note that this states the moon follows an elliptical orbit around earth *and* EMCOM, or vice versa.

This last example of work highlights the process involved with rearranging a formula to solve for a certain variable. This particular function deals with the location of the "tip" or vertex of two lines drawn which are tangent to both the sun and earth and which are symmetrical to each other. This work occurred while experimenting with the idea of incorporating eclipse predictions with the ephemeris data.



Appendix III Orbital Calculator Code

Screen setup

sync on

sync rate 50

Inputs

Input “Start Year: “,inputyear

3D Object Creation and initial setup

make object sphere 1,20

make object sphere 2,14

make object cylinder 3,2

make object sphere 4,60

color object 1,rgb(0,100,0)

color object 3,rgb(255,0,0)

color object 4,rgb(80,50,50)

position object 4,0,0,0

rotate object 3,90,0,0

scale object 3,100,1500,100

position camera 0,0,400

Noting Obital Parameters

remstart

J2000 default values

EMCOM:

Semimajor axis (106 km) 149.60

Sidereal orbit period (days) 365.256

Perihelion (106 km) 147.09

Aphelion (106 km) 152.10

Mean orbital velocity (km/s) 29.78

Max. orbital velocity (km/s) 30.29

Min. orbital velocity (km/s) 29.29

Orbit eccentricity 0.0167

Sidereal rotation period (hrs) 23.9345

Length of day (hrs) 24.0000

Obliquity to orbit (deg) 23.45

MOON:

Semimajor axis (106 km) 0.3844

Perigee (106 km)* 0.3633

Apogee (106 km)* 0.4055

Revolution period (days) 27.3217

Synodic period (days) 29.53

Mean orbital velocity (km/s) 1.023

Max. orbital velocity (km/s) 1.076

Min. orbital velocity (km/s) 0.964

Inclination to ecliptic (deg) 5.145

Inclination to equator (deg) 18.28 – 28.58

Orbit eccentricity 0.0549

Sidereal rotation period (hrs) 655.728

Obliquity to orbit (deg) 6.68

Recession rate from Earth (cm/yr) 3.8 (.000038 km)

remend

Setup Calculations

`inputyear=0

yeardiff=inputyear-2000

earthperiod#=365.256

lunarperiod#=27.3217

lunarrevolution#=earthperiod#/lunarperiod#

interval#=360/earthperiod#

EMCOMradius#=148.500000

Moonradius#=33.84400

moonrecessionvalue#=0.000038

earthrecessionvalue#=(0.000038/81.3)

`ID#=247.98

year=0

Start Do Loop

do

Year Calculation Functions

currentyear=year+inputyear

yeardiff=currentyear-2000

moonrecession#=moonrecessionvalue#*yeardiff

earthrecession#=earthrecessionvalue#*yeardiff

Interval Calculation

`ID#=ID#+0.000001

`ID#=1080

ID#=ID#+interval#

IZ#=5.145*Cos(ID#*lunarrevolution#)

Radius Calculations

NUMEMCOM#=EMCOMradius#*(1-(0.0167*0.0167))

DENEMCOM#=1+((0.0167)*(COS(ID#)))

NUMM#=Moonradius#*(1-(0.0549*0.0549))

DENM#=1+(0.0549*COS(ID#))

REMCOM#=NUMEMCOM#/DENEMCOM#

1RM#=NUMM#/DENM#

RE#=1RM#-(((5.9736*1000)/((5.9736*1000)+(7.349*10)))*1RM#)

2RM#=1RM#-RE#

Horizontal Radius Calculations

HRE#=RE#*cos(IZ#)

HRM#=2RM#*cos(IZ#)

X, Y, and Z Position Calculations

XEMCOM#=REMCOM#*Cos(ID#)

YEMCOM#=REMCOM#*Sin(ID#)

XE#=HRE#*Cos(ID#*lunarrevolution#)

YE#=HRE#*Sin(ID#*lunarrevolution#)

if IZ#>0

```
ZE#=sqrt((RE#*RE#)-(HRE#*HRE#))
endif
if IZ#<0
ZE#=-1*sqrt((RE#*RE#)-(HRE#*HRE#))
endif
XM#=HRM#*(-1*Cos(ID#*lunarrevolution#))
YM#=HRM#*(-1*Sin(ID#*lunarrevolution#))
if IZ#>0
ZM#=sqrt((2RM#*2RM#)-(HRM#*HRM#))
endif
if IZ#<0
ZM#=-1*sqrt((2RM#*2RM#)-(HRM#*HRM#))
endif
```

Positioning 3D Objects

```
Position object 1,XEMCOM#+XE#,YEMCOM#+YE#,ZE#
Position object 2,XEMCOM#+XM#,YEMCOM#+YM#,ZM#
Position object 3,XEMCOM#,YEMCOM#,0
```

Data Printing on Screen

```
Set cursor 0,0
print “Earth X Position: “,XE#
print “Earth Y Position: “,YE#
print “Moon X Position: “,XM#
print “Moon Y Position: “,YM#
```

print “Interval: “,ID#

print “Z Interval: “,IZ#

print “Year: “,year

print “Year difference: “,yeardiff

print “Current Year: “,currentyear

print “EMCOMradius: “,EMCOMradius#

print “Moonradius: “,Moonradius#

print earthrecession#

print moonrecession#

Camera Position and Focus Point

`position camera XEMCOM#+XE#,YEMCOM#+YE#,100

`point camera XEMCOM#+XE#,YEMCOM#+YE#,0

position camera 0,0,300

point camera 0,0,0

Recession Calculation

if ID#<1

EMCOMradius#=148.5+earthrecession#

Moonradius#=33.844+Moonrecession#

endif

if ID#<=360*(year+1) and ID#>=359*(year+1)

year=year+1

EMCOMradius#=148.5+earthrecession#

Moonradius#=33.844+Moonrecession#

endif

End Loop

sync

loop

Appendix IV User Interface Code

```
`Window Setup

`set window on

`set window layout 0, 0, 0

`set window position 220,140


`System Conditioning

sync on

sync rate 50

set camera range 1,1000000000


`Environment Setup

backdrop on

color backdrop rgb(150,150,150)


`Load Images

load image "mainbar.bmp",1


load image "filenorm.bmp",3

load image "filehi.bmp",4

load image "filedep.bmp",5


load image "setdatanorm.bmp",6
```

```
load image "setdatahi.bmp",7
load image "setdatadep.bmp",8

load image "restartnorm.bmp",9
load image "opennorm.bmp",10
load image "savenorm.bmp",11
load image "writenorm.bmp",12
load image "exitnorm.bmp",13

load image "restarthi.bmp",14
load image "openhi.bmp",15
load image "savehi.bmp",16
load image "writehi.bmp",17
load image "exithi.bmp",18

load image "currentnorm.bmp",19
load image "customnorm.bmp",20
load image "currenthi.bmp",21
load image "customhi.bmp",22

load image "playpauseplaying.bmp",23
load image "PlayPauseSelect.bmp",24
load image "PlayPause.bmp",25
```

```
load image "varboxsunhi.bmp",26
load image "varboxearhi.bmp",27
load image "varboxmoohi.bmp",28

load image "varboxsundep.bmp",29
load image "varboxeardep.bmp",30
load image "varboxmoodep.bmp",31

make object sphere 1,12742*800
make object sphere 2,3475*2000
make object cylinder 3,4000*2000
make object sphere 4,1392000*10

color object 1,rgb(0,100,0)
color object 3,rgb(255,0,0)
color object 4,rgb(80,50,50)

position object 4,0,0,0
scale object 3,10,150,10

  `x axis
  make object cylinder 12,1392000*20
```

```
  scale object 12,2,200,2
  rotate object 12,90,0,0
  color object 12,rgb(0,0,255)

  `y axis
  make object cylinder 11,1392000*20
  scale object 11,2,200,2
  color object 11,rgb(0,255,0)

  `z axis
  make object cylinder 10,1392000*20
  scale object 10,2,200,2
  rotate object 10,0,0,90
  color object 10,rgb(255,0,0)

`Assign constant varible values
  `Interface varibles
    mdown=-1
    mdown2=-1
    azimuth#=300
    camdist#=350000000
    paused=0
    sunselect=1
```

```
    datenumber=5
  `Model Varibles

inputyear=2000
yeardiff=inputyear-2000
earthperiod#=365.256
lunarperiod#=27.3217
lunarrevolution#=earthperiod#/lunarperiod#
interval#=360/earthperiod#
EMCOMradius#=148500000
Moonradius#=384400
moonrecessionvalue#=0.000038
earthrecessionvalue#=(0.000038/81.3)
correctionangle#=19
ID#=100.4
year=0

`Sets up the sun-earth-moon system

currentyear=year+inputyear
yeardiff=currentyear-2000
moonrecession#=moonrecessionvalue#*yeardiff
earthrecession#=earthrecessionvalue#*yeardiff
```

ID#=0

Day=Day+1

IZ#=5.145*Cos(ID#*lunarrevolution#)

NUMEMCOM#=EMCOMradius#*(1-(0.0167*0.0167))

DENEMCOM#=1+((0.0167)*(COS(ID#)))

NUMM#=Moonradius#*(1-(0.0549*0.0549))

DENM#=1+(0.0549*COS(ID#))

REMCOM#=NUMEMCOM#/DENEMCOM#

1RM#=NUMM#/DENM#

RE#=1RM#-(.9878*1RM#)

2RM#=1RM#-RE#

HRE#=RE#*cos(IZ#)

HRM#=2RM#*cos(IZ#)

XEMCOM#=(REMCOM#*Cos(ID#))+183377.442

YEMCOM#=(REMCOM#*Sin(ID#))-1860044.48

XE#=HRE#*Cos(ID#*lunarrevolution#+correctionangle#)

YE#=HRE#*Sin(ID#*lunarrevolution#+correctionangle#)

if IZ#>0

```
ZE#=sqrt((RE#*RE#)-(HRE#*HRE#))
endif
if IZ#<0
ZE#=-1*sqrt((RE#*RE#)-(HRE#*HRE#))
endif

XM#=HRM#*(-1*Cos(ID#*lunarrevolution#+correctionangle#))
YM#=HRM#*(-1*Sin(ID#*lunarrevolution#+correctionangle#))

if IZ#>0
ZM#=sqrt((2RM#*2RM#)-(HRM#*HRM#))
endif
if IZ#<0
ZM#=-1*sqrt((2RM#*2RM#)-(HRM#*HRM#))
endif

Position object 1,XEMCOM#+XE#,ZE#,YEMCOM#+YE#
Position object 2,XEMCOM#+(100*XM#),ZM#,YEMCOM#+(100*YM#)
Position object 3,XEMCOM#,0,YEMCOM#

if ID#<1
EMCOMradius#=148500000+earthrecession#
Moonradius#=384400+Moonrecession#
```

```
endif

if ID#<=360*(year+1) and ID#>=359*(year+1)
year=year+1
EMCOMradius#=148500000+earthrecession#
Moonradius#=384400+Moonrecession#
endif

`Sets The Camera Up
camx# = camdist# * sin(azimuth#) * cos(bearing#)
camz# = camdist# * sin(azimuth#) * sin(bearing#)
camy# = camdist# * cos(azimuth#)
position camera camx#,camy#,camz#
point camera 0,0,0

do
`File Button
if filedepr=1
sprite 4,0,0,5

`Textures and provides the functionality for the reset button
if mousex()>0 and mousex()<115 and mousey()>31 and mousey()<49
sprite 9,sprite x(9),sprite y(9),14
```

```
if mouseclick()=1
ID#=0
endif
else
if sprite exist(9)=1
sprite 9,sprite x(9),sprite y(9),9
endif
endif

`Textures and provides the functionality for the open button
if mousex()>0 and mousex()<115 and mousey()>53 and mousey()<71
sprite 10,sprite x(10),sprite y(10),15
if mouseclick()=1
openfile=1
endif
else
if sprite exist(10)=1
sprite 10,sprite x(10),sprite y(10),10
endif
endif

`The actual funtionality for the open button, it reads the file and then assigns variables
for the given date
```

```
if openfile=1
if tipe$(0)<>" " and opencleared=0
tipe$(1)
opencleared=1
endif
tipe$(0)
if returnkey()=1
open to read 1,tipe$(0)
read string 1,date$
readdate=val(date$)
close file 1
openfile=0
openday$=left$(readdate$,4)
openday=val(right$(openday$,2))
openmonth=val(left$(readdate$,2))
openyear=val(right$(readdate$,4))
ID#=0
openfile=0
opencleared=0
endif
endif
```

`Textures and provides the functionality for the save button

```
if mousex()>0 and mousex()<115 and mousey()>76 and mousey()<93
sprite 11,sprite x(11),sprite y(11),16
if mouseclick()=1
savefile=1
endif
else
if sprite exist(11)=1
sprite 11,sprite x(11),sprite y(11),11
endif
endif

`Creates a file based on user input as controlled by tipe$() then writes the time data to
the file
`(this is still part of the save button)
if savefile=1
if tipe$(0)<>" " and savecleared=0
tipe$(1)
savecleared=1
endif
tipe$(0)
if returnkey()=1
open to write 1,tipe$(0)
write string 1,str$(01012000)
```

```
close file 1
savefile=0
savecleared=0
endif
endif

`Textures and provides the functionality for the Writedata button
if mousex()>0 and mousex()<115 and mousey()>98 and mousey()<115
sprite 12,sprite x(12),sprite y(12),17
if mouseclick()=1
writefile=1
endif
else
if sprite exist(12)=1
sprite 12,sprite x(12),sprite y(12),12
endif
endif

`Creates a file based on user input as controlled by tipe$() then writes the time data to
the file
`in a human readable form
`(this is still part of the save button)
if writefile=1
```

```
if tipe$(0)<>" " and writecleared=0
tipe$(1)
writecleared=1
endif
tipe$(0)
if returnkey()=1
writefile$=tipe$(0)+".txt"
open to write 1,writefile$
write string 1,"-------------------------------------------------------------------------------"
write string 1,"Ephemeris Data for Date# to Date# Body: (Earth/Moon/Emcom)
|"
write string 1,"-------------------------------------------------------------------------------"
write string 1,"Heliocentric Vector-type ephemeris data                              |"
write string 1,"-------------------------------------------------------------------------------"
write string 1,"Date      |X          |Y          |Z          |Angle        |Radius          |"
write string 1,"-------------------------------------------------------------------------------"
for writestring=1 to datenumber
write string 1,"          |           |           |           |             |                |"
next writestring
write string 1,"-------------------------------------------------------------------------------"
write string 1,"March 2009. John, Powell, Ted"
close file 1
writefile=0
```

```
writecleared=0
endif
endif

`Textures and provides the functionality for the exit button
if mousex()>0 and mousex()<115 and mousey()>120 and mousey()<138
sprite 13,sprite x(13),sprite y(13),18
if mouseclick()=1
end
endif
else
if sprite exist(13)=1
sprite 13,sprite x(13),sprite y(13),13
endif
endif

endif

`This provides the "depression" control and creates the gradual sliding effect observed
in the menus
if mousex()>0 and mousex()<61 and mousey()>0 and mousey()<17 and filedepr=1 and
mouseclick()=2
filedepr=0
```

```
endif

if mousex()>0 and mousex()<61 and mousey()>0 and mousey()<17 and filedepr=0

`This specific part controls the "downslide of the file menu"
if c<50
c=c+4
endif

for b=9 to 13
sprite b,0,(((c)-(((b-8)*(c*-.85))-5))/2)-20,b
next b

sprite 4,0,0,4

`This is where the "depression is set" by clicking the mouse
if mouseclick()=1 and c>50 or c<0
filedepr=1
endif

else

`This specific part provides the "upslide of the file menu"
```

```
if filedepr=0
if c>0
c=c-4
endif

for e=9 to 13
sprite e,0,(((c)-(((e-8)*(c*-.85))-5))/2)-20,e
next e
sprite 4,0,0,3

endif
endif

`This next part controls the set data button, much the same as the file button
if optidepr=1

sprite 5,61,0,7

`This controls set current date, which sets the date based of your computer
if mousex()>120 and mousex()<235 and mousey()>29 and mousey()<48
sprite 19,sprite x(19),sprite y(19),21
if mouseclick()=1
ID#=0
```

```
endif
else
if sprite exist(19)=1
sprite 19,sprite x(19),sprite y(19),19
endif
endif

`This controls the set custom date, allowing for user input via a modified tipe$() funtion
if mousex()>120 and mousex()<235 and mousey()>51 and mousey()<70
sprite 20,sprite x(20),sprite y(20),22
if mouseclick()=1
setcustomdate=1
endif
else
if sprite exist(20)=1
sprite 20,sprite x(20),sprite y(20),20
endif
endif

`This is where the actual funtionality of the set custom date takes place
if setcustomdate=1

`Allows for custom day
```

```
if daydone=0 and monthdone=1 and yeardone=1
set cursor 0,125
print "Enter Day:"
tipe(0)
if returnkey()=1 and tipe(0)>0
tipeday=tipe(0)
daydone=1
if tipe(0)<>0 and yearcleared=1 and monthcleared=1 and daycleared=0
tipe(1)
daycleared=1
endif
endif
endif

`Allows for custom month
if daydone=0 and monthdone=0 and yeardone=1
set cursor 0,125
print "Enter Month:"
tipe(0)
if returnkey()=1 and tipe(0)>0
tipemonth=tipe(0)
monthdone=1
if tipe(0)<>0 and yearcleared=1 and monthcleared=0 and daycleared=0
```

```
tipe(1)
monthcleared=1
endif
endif
endif

`Allows for custom year
if daydone=0 and monthdone=0 and yeardone=0
set cursor 0,125
print "Enter Year:"
tipe(0)
if returnkey()=1 and tipe(0)>0
tipeyear=tipe(0)
yeardone=1
if tipe(0)<>0 and yearcleared=0 and monthcleared=0 and daycleared=0
tipe(1)
yearcleared=1
endif
endif
endif

`Sets the rotational data based off of the custom input variables
if daydone=1 and monthdone=1 and yeardone=1
```

```
ID#=0
setcustomdate=0
endif

endif
endif

`This provides the "depression" control and creates the gradual sliding effect observed
in the menus(for the
`data menu
if mousex()>61 and mousex()<123 and mousey()>0 and mousey()<17 and optidepr=1
and mouseclick()=2
optidepr=0
endif

`controls gradual sliding for the data menu
if mousex()>61 and mousex()<123 and mousey()>0 and mousey()<17 and optidepr=0

if d<50
d=d+4
endif

for f=19 to 20
```

```
sprite f,120,(((d)-(((f-18)*(d*-.85))-5))/2)-22,f
next f

sprite 5,61,0,8

if mouseclick()=1 and d>50 or d<0
optidepr=1
endif

else

if optidepr=0
if d>0
d=d-4
endif

for g=19 to 20
sprite g,120,(((d)-(((g-18)*(d*-.85))-5))/2)-22,g
next g
sprite 5,61,0,6

endif
endif
```

```
`Fills in the rest of the file bar
sprite 1,123,0,1

`Orbital calc as designed by Powell and Ted
set cursor 200,200
print converter$(0)
if paused=0
set cursor 200,200
print converter$(1)

currentyear=year+inputyear
yeardiff=currentyear-2000
moonrecession#=moonrecessionvalue#*yeardiff
earthrecession#=earthrecessionvalue#*yeardiff

ID#=ID#+interval#
Day=Day+1
IZ#=5.145*Cos(ID#*lunarrevolution#)

NUMEMCOM#=EMCOMradius#*(1-(0.0167*0.0167))
DENEMCOM#=1+((0.0167)*(COS(ID#)))
NUMM#=Moonradius#*(1-(0.0549*0.0549))
```

DENM#=1+(0.0549*COS(ID#))

REMCOM#=NUMEMCOM#/DENEMCOM#

1RM#=NUMM#/DENM#

RE#=1RM#-(.9878*1RM#)

2RM#=1RM#-RE#

HRE#=RE#*cos(IZ#)

HRM#=2RM#*cos(IZ#)

XEMCOM#=(REMCOM#*Cos(ID#))+183377.442

YEMCOM#=(REMCOM#*Sin(ID#))-1860044.48

XE#=HRE#*Cos(ID#*lunarrevolution#+correctionangle#)

YE#=HRE#*Sin(ID#*lunarrevolution#+correctionangle#)

if IZ#>0

ZE#=sqrt((RE#*RE#)-(HRE#*HRE#))

endif

if IZ#<0

ZE#=-1*sqrt((RE#*RE#)-(HRE#*HRE#))

endif

XM#=HRM#*(-1*Cos(ID#*lunarrevolution#+correctionangle#))

YM#=HRM#*(-1*Sin(ID#*lunarrevolution#+correctionangle#))

```
if IZ#>0
ZM#=sqrt((2RM#*2RM#)-(HRM#*HRM#))
endif
if IZ#<0
ZM#=-1*sqrt((2RM#*2RM#)-(HRM#*HRM#))
endif

Position object 1,XEMCOM#+XE#,ZE#,YEMCOM#+YE#
Position object 2,XEMCOM#+(100*XM#),ZM#,YEMCOM#+(100*YM#)
Position object 3,XEMCOM#,0,YEMCOM#

if ID#<1
EMCOMradius#=148500000+earthrecession#
Moonradius#=384400+Moonrecession#
endif

if ID#<=360*(year+1) and ID#>=359*(year+1)
year=year+1
EMCOMradius#=148500000+earthrecession#
Moonradius#=384400+Moonrecession#
endif
endif
```

```
`the end of the orbital calculator


`Controls the variable box, (Display of both the variables and the box itsself)
if sprite exist(27)
hide sprite 27
endif


`Controls the sun button
if mousex()>0 and mousex()<47 and mousey()>342 and mousey()<353
sprite 27,1,340,26
show sprite 27
if mouseclick()=1
sunselect=1
earselect=0
mooselect=0
endif
endif


`Controls the Earth Button
if mousex()>48 and mousex()<92 and mousey()>342 and mousey()<353
sprite 27,47,340,27
show sprite 27
if mouseclick()=1
```

```
sunselect=0
earselect=1
mooselect=0
endif
endif

`Controls the Moon button
if mousex()>93 and mousex()<137 and mousey()>342 and mousey()<353
sprite 27,92,340,28
show sprite 27
if mouseclick()=1
sunselect=0
earselect=0
mooselect=1
endif
endif

`Displays variable information
if sunselect=1
set cursor 5,350
print "a sun variable"
sprite 26,1,340,29
endif
```

```
if earselect=1
set cursor 5,350
print "a earth variable"
sprite 26,1,340,30
endif
if mooselect=1
set cursor 5,350
print "a moon variable"
sprite 26,1,340,31
endif

`Controls the PlayPause button

if reseter=1 and mouseclick()=1
reseter=0
endif

if mousex()>164 and mousex()<204 and mousey()>427 and mousey()<466 and
paused=0
sprite 15,150,413,24
if mouseclick()=1
paused=1
reseter=1
```

```
endif
else
sprite 15,150,413,23
endif

if paused=1
sprite 15,150,413,25
endif

if paused=1 and mousex()>164 and mousex()<204 and mousey()>427 and
mousey()<466 and mouseclick()=1 and reseter=0
paused=0
endif



`Provides Funtionability for the 3-D camera movements
if camerabutton()=1

`camdist#=500
movex=mousemovex()
movey=mousemovey()
azimuth# = azimuth# + movey/3
```

```
bearing# = wrapvalue(bearing# + movex/3)
if azimuth#<181 then azimuth#=181
if azimuth#>359 then azimuth#=359
camx# = camdist# * sin(azimuth#) * cos(bearing#)
camz# = camdist# * sin(azimuth#) * sin(bearing#)
camy# = camdist# * cos(azimuth#)
position camera camx#,camy#,camz#
point camera 0,0,0

endif

sync
loop

`This funtion allows for the rightclick
function camerabutton()

if locked=1 and mouseclick()=1
locked=0
show mouse
endif

if mouseclick()=2 and locked=0 and mousey()>17
```

```
locked=1
msx#=mousex()
msy#=mousey()
endif

if locked=1
position mouse msx#,msy#
hide mouse
endif

endfunction locked

function tipe$(cleartipe)
if cleartipe=1
filename$=""
else
endif

if fr=0
fr=1
dim letter$(25)
letter$(0)="a"
letter$(1)="s"
```

```
letter$(2)="d"
letter$(3)="f"
letter$(4)="g"
letter$(5)="h"
letter$(6)="j"
letter$(7)="k"
letter$(8)="l"
letter$(9)="z"
letter$(10)="x"
letter$(11)="c"
letter$(12)="v"
letter$(13)="b"
letter$(14)="n"
letter$(15)="m"
letter$(16)="q"
letter$(17)="w"
letter$(18)="e"
letter$(19)="r"
letter$(20)="t"
letter$(21)="y"
letter$(22)="u"
letter$(23)="i"
letter$(24)="o"
```

```
letter$(25)="p"
endif

for character=16 to 25
if scancode()=character and oldscan<>scancode()
oldscan=scancode()
characterstyped=characterstyped+1
nextlettercounter=10
filename$=filename$+letter$(character)
endif
next character

for character2=30 to 38
if scancode()=character2 and oldscan2<>scancode()
oldscan2=scancode()
characterstyped=characterstyped+1
nextlettercounter=10
filename$=filename$+letter$(character2-30)
endif
next character2

for character3=44 to 50
if scancode()=character3 and oldscan3<>scancode()
```

```
oldscan3=scancode()
characterstyped=characterstyped+1
nextlettercounter=10
filename$=filename$+letter$(character3-35)
endif
next character3

if nextlettercounter>0
nextlettercounter=nextlettercounter-1
else
oldscan=0
oldscan2=0
oldscan3=0
endif

if scancode()=14 and characterstyped>0
characterstyped=characterstyped-1
filename$=left$(filename$,characterstyped-1)
endif
set cursor 0,145
print filename$

endfunction filename$
```

```
function tipe(cleartipe)
if cleartipe=1
value=0
value$=" "
endif

if fr=0
fr=1
dim number$(9)

number$(0)="1"
number$(1)="2"
number$(2)="3"
number$(3)="4"
number$(4)="5"
number$(5)="6"
number$(6)="7"
number$(7)="8"
number$(8)="9"
number$(9)="0"
endif
```

```
for character=2 to 11
if scancode()=character and oldscan<>scancode()
oldscan=scancode()
characterstyped=characterstyped+1
nextlettercounter=10
value$=value$+number$(character-2)
endif
next character

if nextlettercounter>0
nextlettercounter=nextlettercounter-1
else
oldscan=0
endif

if scancode()=14 and characterstyped>0
characterstyped=characterstyped-1
value$=left$(value$,characterstyped-1)
endif
set cursor 0,145
print value$
value=val(value$)
endfunction value
```

```
function converter$(running)
if running=1
day#=day#+1
endif
day=day#

if day#>31.0
months=1
day=0
endif

if day#>59.0
months=2
day=0
endif

if day#>90.0
months=3
day=0
endif

if day#>120.0
```

```
months=4
day=0
endif

if day#>151.0
months=5
day=0
endif

if day#>181.0
months=6
day=0
endif

if day#>212.0
months=7
day=0
endif

if day#>243.0
months=8
day=0
endif
```

```
if day#>273.0
months=9
day=0
endif

if day#>304.0
months=10
day=0
endif

if day#>334.0
months=11
day=0
endif

if day#>365.0
day#=0
day=0
year=year+1
endif

returndate$=str$(months)+"/"+str$(day)+"/"+str$(year+2000)
```

```
endfunction returndate$
```