FAST-Graphics, Solar System Simulation for Microcomputers
J. D. Callahan, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109

INTRODUCTION. In order to collect meaningful data of the solar system it is important to plan observations with care. Hastily or poorly planned observations can result in hours of frustration when the observer tries to extract meaningful information from the data. Observers are often limited to "batch" computer programs to generate large volumes of printout giving possible observation times. This printout must be sorted through and some sense made out of it—an often tricky and usually time-consuming process. The observer is often forced to do much guess work. This usually causes the observer to make his "batch" program smarter, but this results in more computer time. These problems are exagerted on a microcomputer, where computer time is limited and the ability to print reams of paper is not desired.

At JPL, over the last 5 years a highly successful interactive graphics planning tool has been written (refs. 1-3). Its name is Mip, for Multimission Interactive Planner. The program simulates the positions and motions of stars and solar system bodies, and can typically generate a complex scene in less than a second on a microcomputer. On minicomputers with fast graphics devices, it is possible to generate several frames per second, providing a near movie effect. The program is highly accurate, versatile, and portable.

Whether as a stand-alone planning tool, or as a reviewer for "batch" planners, Mip has proved to be an innovative and valuable tool. The program runs on many computers at JPL, and it can simulate Earth-based as well as spacecraft-based instruments.

INTERACTIVE INTERFACE. Rather than using a special input device, such as a joy stick or mouse, Mip uses a standard keyboard. At first, it might seem that this would limit the flexibility of an interactive graphics program. However, given the many keys, and with a little creativity, it was possible to design a very effective interface. For example, keys "c" and "v" soon will zoom the field of view in and out, and key "x" will toggle through a series of exact field sizes. Key "n" will change the sensitivity (through 3 levels) of keys "c" and "v", and other keys. Keys "2", "3", and "4" will rotate the field, and so forth.

THEORY. In the following discussion, the expression $|\theta_i|$ is used to denote a $3 \times 3$ rotation transformation matrix about the ith axis by the angle $\theta$. Stellar aberration is not done in Mip, because this usually results in small errors. A light-time correction is made, but, in order to save computation time, the correction value from the previous scene is used for the current scene, unless it has changed by more than 2 seconds. It is important to have a fast way to obtain the positions of solar system bodies, such as Chebyshev polynomial evaluation. Body positions should be done in double precision, but details like grids and landmarks should be done in single precision, to save computation time.

Mip first constructs a transformation matrix from inertial space (in this case B1950) to instrument, or line-of-sight space. Call this matrix $TITV$, for transformation from inertial to TV (instrument). The construction of this matrix depends on the instrument involved, but is usually fairly easy to form. Bodies are rotated to their correct latitude and longitude using the IAU standard, but terms below $1^\circ$ have been dropped. Rotations are done in the standard way, such as $TBFJ = [1 + \alpha - 90^\circ][1 - 90^\circ + \delta][1 + \omega]$, where $TBFJ$ is a $3 \times 3$ transformation matrix from body-fixed to inertial, $\alpha$ is the R.A. of the body pole, $\delta$ is the DEC of the body pole, and $\omega$ is the rotation of the prime meridian.

Landmarks are represented as circles. For large landmarks, the rim of the feature will still be on the body. The points which describe the landmarks on a body (scaled to unit radius) are computed once and then stored within the program. When the landmarks are to be displayed on the screen, they are transformed and scaled along with the grid lines (also computed and stored once on a unit body) to get final picture coordinates.

Bodies are represented as tri-axial ellipsoids by scaling the unit sphere which contains grid lines and landmarks. The scaling matrix is called $TS$, and has as its diagonal $a$, $b$, and $c$, which are the prime, intermediate, and small axis of the tri-axial ellipsoid, respectively. Grid lines are distorted from their true positions, but these lines are only drawn to show the shape of the body. Unless a body is very ellipsoidal the distortion is not appreciable. Landmarks retain their true positions even after the scaling, because a correction is made to the latitude and longitude before the points are stored. The correction follows the formulation ($lt_c$ and $lg_c$ are the corrected latituded and longitude, and are easily solved for)

$$m \begin{bmatrix} \cos(lg) \cos(lt) \\ \sin(lg) \cos(lt) \\ \sin(lt) \end{bmatrix} = TS \begin{bmatrix} \cos(lg_c) \cos(lt_c) \\ \sin(lg_c) \cos(lt_c) \\ \sin(lt_c) \end{bmatrix}$$

with $m$ being the distance from the origin to the ellipsoid.

In order to get an approximate tri-axial limb in TV coordinates, first transform an $x$-$y$ plane circle in TV coordinates to body-fixed coordinates. Call the transformation matrix, $TBFTV^T = (TITV \ TBFJ)^T$. Now that the circle is represented in the body's coordinates, it may be scaled by the axes of the tri-axial ellipsoid. That is, apply the matrix $TS$ described above. Now transform back to the TV system, and the original circle will be properly scaled to give the limb. If we call the set of points representing a circle in TV space, $\hat{c}$, and the limb in TV space, $l$, then the following equation holds:

$$l = TBFTV \ TS \ TBFTV^T \hat{c}.$$  

To get an approximate terminator of a tri-axial body in TV coordinates, do the following. Knowing the sun direction, construct a transformation matrix from a sun system, where the z axis is in the direction of the sun, to inertial space. Now transform an $x$-$y$ plane circle in sun space to inertial space to body-fixed space; scale by the tri-axial body as above, and then transform back to TV space. If we call the circle in sun space $\hat{c}$, the transformation matrix from sun to inertial space $TSI$, the transformation from inertial to body-fixed space, $TBFI^T$, and the terminator in
In order to get the correct camera orientation for the stars, start with TITV. In inertial space, there is a unit vector which points in the direction of the camera. This vector is the z-axis in the TV coordinate system. Therefore, it's coordinates in inertial space are (let the components of TITV be denoted by \(t\)): \((t_1, t_2, t_3)\). Now the projection of this vector onto the x-y inertial plane is: \((t_1, t_2, 0)\). Transforming this vector into TV space by multiplying it by \(TITV\) and taking only the x-y projection in TV space, yields a vector normal to the direction of sight (that is, normal to the z-axis in TV space) and along the direction of increasing or decreasing declination (in other words pointing north or south). Call this vector \(\mathbf{p} = (t_1, t_2, t_3, 0)\).

One may obtain the angle of rotation, \(\theta\) (x-axis in TV coordinates to the p vector), with the components of the p vector and the arctangent function. It is a simple matter then to form the transformation matrix from R.A. and DEC coordinates to TV line-of-sight coordinates. Call this matrix \(TRDTV\), then \(TRDTV = [90° - \theta]p\).

GRAPHICS. Within the FORTRAN 77 code of \(\text{Mip}\), all rotations, clipping, and alphanumeric generation are done. All that is required of the graphics software is to (1) clear the screen, (2) change the color, and (3) connect the 2-D points with straight lines. This makes \(\text{Mip}\) extremely portable. Once final TV coordinates are calculated (x-axis is the line-of-sight), the \(x\) and \(y\) coordinates are scaled by the distance of the observer and then plotted on the 2-D screen. This is a simple orthographic projection.

The resolution to draw bodies is varied by the program automatically depending on the size of the bodies and a user input resolution value. The maximum possible resolution is 1°. The resolution is determined by the equation

\[
\theta = 2\cos^{-1}(1 - D/r)
\]

where \(\theta\) is the resolution angle in degrees, \(r\) is the radius of the body in question (screen units), and \(D\) is a user input resolution value. \(\text{Mip}\) allows 5 levels of resolution. A value of \(D = .7\) will give very satisfactory results — segments rarely being visible. The program picks an integer value of \(\theta\) from the set (divisors of 360): 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 18, 20, 30, 36, 45, 60, 90. The equation above is based on the concept that the maximum difference between a true arc and the actual straight line connecting the points is no greater than the value \(D\).

Once the 2-D points have been calculated, a clipping routine is applied to get the actual points to be plotted. The program is smart enough not to apply this procedure to bodies which are obviously way outside of the field of view.

Hidden line removal is done correctly for bodies as follows. Given a tri-axial body with axes a, b, and c, a vector perpendicular to the surface at \((x, y, z)\) is \(\mathbf{p} = (x/a^2, y/b^2, z/c^2)\). If we call the line-of-sight vector \(\mathbf{t}\), then \(\mathbf{p} \cdot \mathbf{t} < 0\) for the point to be visible. All longitude and latitude (equator only) lines on bodies are composed of one or two semi-circle arcs respectively. \(\text{Mip}\) checks the end-point of an arc to see if it's visible. If it's not, the program checks the other end, and connects points along the arc until they are no longer visible.

Alphanumeric generation is done by simply storing a set of points which describe the letters A-Z, and the digits 0-9. When an alphanumeric character is desired, its coordinates are generated in the proper place by adding the stored coordinates to the location desired.

VIDEO OUTPUT. To the right is an example of the video output one can expect when running \(\text{Mip}\). The view is from the Earth, looking at Mars. Note the landmarks on the surface of Mars, Phobos and Deimos, and a star. The sub-solar point is the triangle, and the terminator has short lines connected to it on the sun side.

References: (1) J. D. Callahan (1982), JPL IOM 314.8-357; (2) J. D. Callahan (1984), JPL IOM 314.8-500; (3) J. D. Callahan (1986), JPL IOM 314.8-651.