Io Java Applet - Display and de-projection of the Io map onto a sphere

Shui Hung Kwok[1]
W. M. Keck Observatory

The Challenge

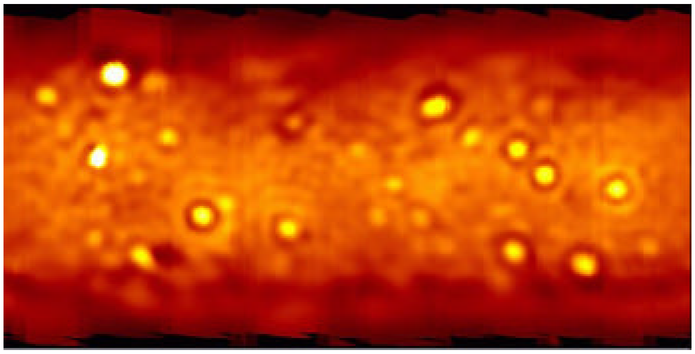

Figure 1: 360° map of Io

Given a 360° map of the Jovian moon Io, figure 1, which is a collage of de-projected images of Io, the challenge was to project the map onto a sphere and to display it as viewing from different angles. The original idea was to produce a sequence of images representing different viewpoints of Io, and to compile them into a movie. Although possible, this idea did not find any acceptance at the Keck Observatory due to lack of necessary tools to create the intermediate images. Instead, James Bartlett, a visiting collaborator from Victoria, BC, Canada, undertook another approach. Using a commercial animation tool, he created several movies depicting Io traveling around the planet Jupiter. He used a method called texture mapping, where the Io image is wrapped around a sphere. The animation software calculates the viewpoint for each frame and projects the image onto the sphere according to the viewing angle. The result is then saved in a popular movie file format, which requires a common media player to view the animation, figure 2.

Once the movie is created, one has little control over how it is played. The rotation speed and the smoothness of transition between frames depend on the number of frames, i.e. intermediate images that are generated and stored in the movie. The more images, the smoother the transition, but it also would decrease the rotation speed. Moreover , it is not possible to change the viewpoint randomly. User interactions are limited to start and stop the movie and to move to a certain frame.

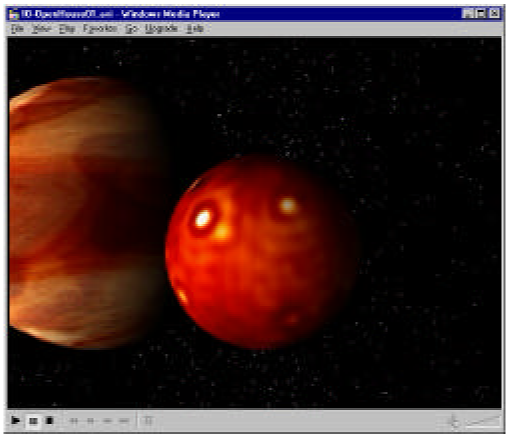

Figure 2: Io movie with Jupiter in the background

[1] skwok@keck.hawaii.edu ; tel 808-885-7887; fax 808-885-4464; http://www.keckobservatory.org; California Association for Research in Astronomy, W.M. Keck Observatory, 65-1120 Mamalahoa Hwy., Kamuela, HI 96743, USA

The Java Solution

A customized Java Applet was developed to provide an interactive illustration of Io. A Java based interactive visualization tool that allows random viewpoint changes and 3D rotation can give users a more satisfactory experience than just viewing an animation. Additional close-up images and a reference map can further enhance the quality of the visualization.

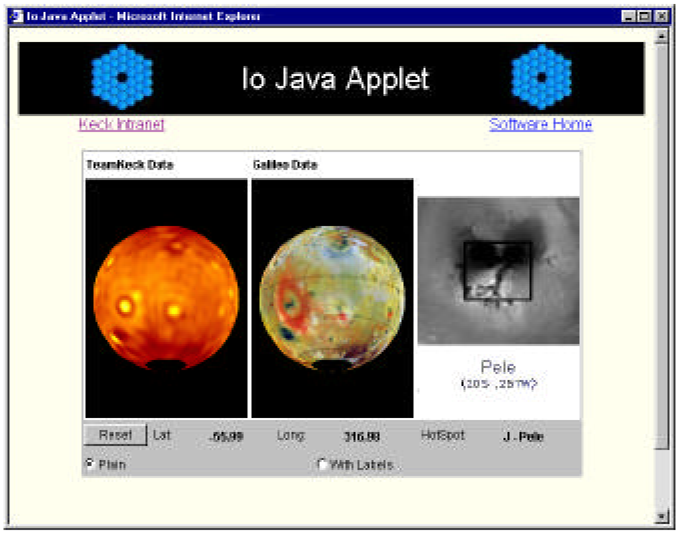

Figure 3: Java Io Applet within a web-browser

The Io Java Applet can run embedded in a web page or as a standalone application. Figure 3 shows the Java Io Applet within a browser window. Figure 4 depicts the same Java program being executed as a standalone application.

The applet displays three images. The first leftmost image represents Io using Team Keck's data. The second image in the center depicts Io with a map created with data from NASA's spacecraft Galileo. The third section shows a close-up image corresponding to the area under the mouse pointer.

User interface is kept to a minimum. There are no menus or pop-up windows. The user moves the mouse pointer over one of the Io images. When the mouse pointer crosses over an area that has been identified, the corresponding close-up image is displayed on the far right section. To rotate the Io images, the user drags the mouse while pressing a mouse button. The rotation mimics the movement of the mouse in both horizontal and vertical directions. While the horizontal rotation is continuous, i.e. it wraps around, there is a maximum tilt limiting the vertical rotation to avoid displaying the images upside down, which can be very confusing for the observer.

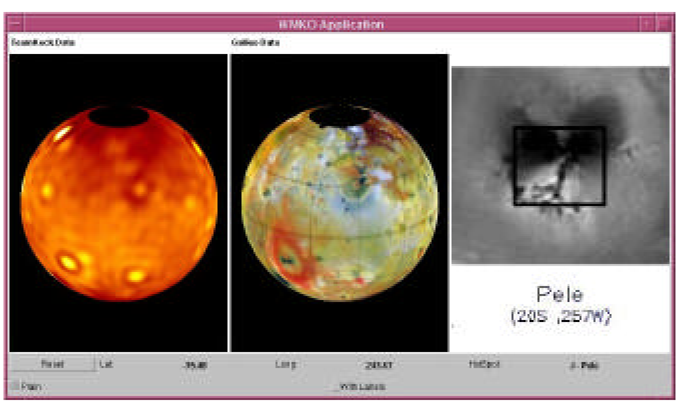

Figure 4: Java Io standalone application

The heart of the Java Io Applet is the specially designed de-projection algorithm that translates the pixels from the two-dimensional map onto the surface of a sphere. Instead of generating intermediate images for different viewpoints as in the movie making process, the applet calculates the position of each pixel for the entire image. In order to achieve an acceptable performance in real-time, an extremely powerful algorithm was developed. In fact, the algorithm is implemented 100% in Java without the use of specialized graphics libraries, which would require the user to download additional Java modules. Java programs are in general slower than other programs, for example programs written in C, because Java code is interpreted by the Java virtual machine and not executed directly by the CPU as native instruction code. To speed up the applet, several acceleration techniques were combined.

First, to avoid repetitive use of slow trigonometric functions, a lookup table is created that contains the translation from screen pixels to position on the sphere. This table is calculated only once and does not need to be updated unless the display window is resized.

Second, the 2D Io input image is converted to a cylindrical projection, where the horizontal axis corresponds to longitude and the vertical axis corresponds to latitude. In the case of the Io map, it is already in cylindrical projection, so no project is required. However, the Galileo's map is in a different projection, which needs to be mapped to a cylindrical projection. This needs to be done only once, when the images are loaded into the program.

Third, for each visible pixel on the screen, it is either a background pixel or there is a corresponding position on sphere. Viewing the sphere from a different angle means that this position needs to be rotated. The result of the rotation is projected back onto the screen to obtain the screen coordinates. This is simply taking the x and y components and dropping the z coordinate. With these new screen coordinates, one can find the new latitude and longitude and therefore the x, and y coordinates on the 2D map. Its corresponding pixel is then copied to the screen.

The present algorithm effectively maps a pixel on the screen to one pixel on the image. With the exception of the polar regions, where severe distortions occur, the mapping covers the entire image leaving no gaps between pixels. Also, only pixels that are displayed are mapped. The algorithm is fast and simple. The only issue is the accuracy of trigonometric functions near 90°, which can be avoided by rotating an additional 90° and subtract 90° from the map coordinates.

Conclusion

The first implementation of the Java Io applet was completed in one week. Testing and additional improvements required approximately 10 days. The effort is small compared to the total effort needed to gather the raw data and to create the Io map. In summary, the Io Java Applet is a simple and effective visualization tool that offers users a fun way to understand and interact with scientific results.