

Real-time Visualization of Spacecraft Telemetry for the GLAST and LRO Missions

Eric T. Stoneking¹, Neerav Shah², and Dean J. Chai³
Goddard Space Flight Center, Greenbelt, MD, 20771

GlastCam and LROCam are closely-related tools developed at NASA Goddard Space Flight Center for real-time visualization of spacecraft telemetry, developed for the Gamma-Ray Large Area Space Telescope (GLAST) and Lunar Reconnaissance Orbiter (LRO) missions, respectively. Derived from a common simulation tool, they use related but different architectures to ingest real-time spacecraft telemetry and ground predicted ephemerides, and to compute and display features of special interest to each mission in its operational environment. We describe the architectures of GlastCam and LROCam, the customizations required to fit into the mission operations environment, and the features that were found to be especially useful in early operations for their respective missions. Both tools have a primary window depicting a three-dimensional “Cam” view of the spacecraft that may be freely manipulated by the user. The scene is augmented with fields of view, pointing constraints, and other features which enhance situational awareness. Each tool also has another “Map” window showing the spacecraft’s groundtrack projected onto a map of the Earth or Moon, along with useful features such as the Sun, eclipse regions, and TDRS satellite locations. Additional windows support specialized checkout tasks. One such window shows the star tracker fields of view, with tracking window locations and the mission star catalog. This view was instrumental for GLAST in quickly resolving a star tracker mounting polarity issue; visualization made the 180-deg mismatch immediately obvious. Full access to GlastCam’s source code also made possible a rapid coarse star tracker mounting calibration with some “on the fly” code adjustments; adding a fine grid to measure alignment offsets, and introducing a calibration quaternion which could be adjusted within GlastCam without perturbing the flight parameters. This calibration, from concept to completion, took less than half an hour. Both GlastCam and LROCam were developed in the C language, with non-proprietary support libraries, for ease of customization and portability. This “no-black-boxes” aspect enables engineers to adapt quickly to unforeseen circumstances in the intense operations environment. GlastCam and LROCam were installed on multiple workstations in the operations support rooms, allowing independent use by multiple subsystems, systems engineers and managers, with negligible draw on telemetry system resources.

I. Introduction

It is trite to say that a picture is worth a thousand words. If ever there were a place to drive home the truth of it, though, that place is a spacecraft operations center. Traditional telemetry displays require some familiarity and interpretation to answer common questions about the spacecraft’s situation: Is the spacecraft in daylight? Is it pointing properly? Is the Earth blocking the star trackers? The advantages of visualization are obvious. So why aren’t we all using visualization in spacecraft operations? There are a few commercial visualization software packages, one of them being the Satellite Tool Kit Visualization Option from Analytical Graphics¹. There are, however, intrinsic limitations to a commercial solution, which boil down to two things. First, it is prohibitively expensive to put an independent instance on every workstation. Second, the proprietary nature of such tools can make it difficult to add or customize features without some technical support from the vendors. The alternative is to

¹ Aerospace Engineer, Code 591. AIAA Member.

² Aerospace Engineer, Code 591. AIAA Senior Member.

³ Aerospace Engineer, Code 591.

develop your own visualization tool from scratch, which may seem daunting. The purpose of this paper is to encourage the reader in considering this alternative. We, the authors, are all GNC engineers, with journeyman knowledge of the C language and some hobby-level familiarity with OpenGL; we are not software engineers. What we do have is a unique familiarity with our respective spacecraft that, when leveraged with OpenGL's graphics, allowed us to develop a customized visualization tool to suit our own needs. In this paper we describe GlastCam and LROCam, the visualization tools we developed from freely-available resources and a few man-months of our time. We describe our experiences with them in support of GLAST and LRO on-orbit checkout operations.

For conciseness in the development description below, we will primarily describe GlastCam, which was developed first. LROCam inherited many of the features of GlastCam, and added a few of its own. Custom display features will be described in separate subsections below.

II. Architecture of the Visualization Software

The visualization process is composed of three functional blocks (See Figure 1). First, the telemetry must be ingested from the telemetry system and auxiliary data files. Second, the telemetry must be processed to derive items of interest. And third, the results must be graphically displayed. This sequence of functions is executed within an event loop, managed by an event handler, which allows user interaction via keyboard and mouse.

Each workstation in the Mission Operations Center (MOC) hosts an application called the data point server (DPS), which receives telemetry from the telemetry network and passes it along to the traditional telemetry pages. This is the primary data source for GlastCam. On initialization, GlastCam opens a socket connection to the DPS, and sends a text message identifying the telemetry points it needs. Thereafter, the DPS sends a text message through the socket at regular intervals, with formatted telemetry data. GlastCam scans this message, mapping the telemetry obtained into internal variables for further processing. This function is straightforward, but requires some techniques not previously encountered in our programming experience. We needed a few hours of assistance from a real software engineer with MOC network expertise to learn how to establish a socket connection. Once it is established, writing to and reading from a socket is just like writing to and reading from a file.

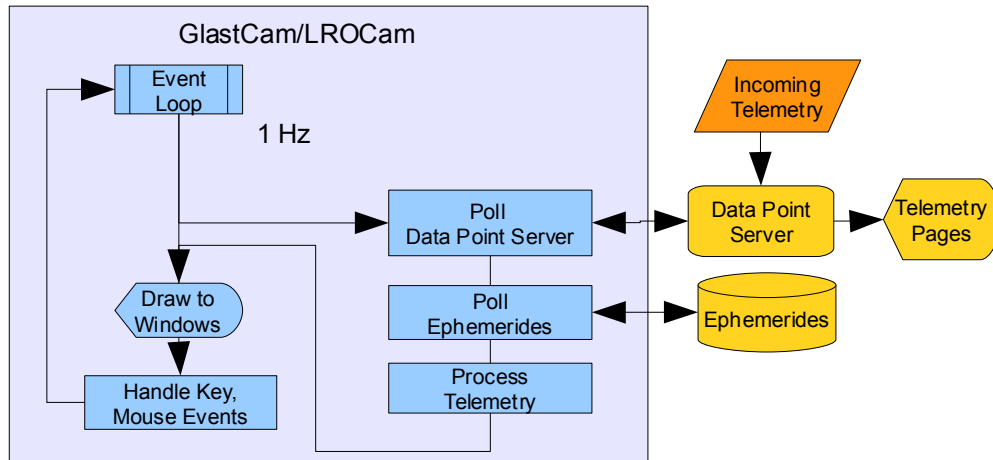


Figure 1: GlastCam/LROCam Block Diagram

After the telemetry points have been mapped in, the second functional block processes the data as needed. In addition to straightforward unit conversions and coordinate transformations, this is also the place to compute ancillary items used for comparison and validation. For example, we used coarse sun sensor telemetry to compute a Sun-pointing vector in the spacecraft body frame. We also used a solar ephemeris, the spacecraft's orbit ephemeris (from telemetry or auxiliary sources) and the attitude telemetry to compute a predicted Sun-pointing vector, also in the spacecraft body frame. Computing the same vector through two independent means provides a valuable consistency check. Aside from general situational awareness, these consistency checks were a key part of our purpose in writing GlastCam.

The most accurate GLAST ground track is based on the real-time GPS telemetry, but this telemetry is only available during contacts. During non-contact periods, GlastCam uses the predicted ephemerides provided by Goddard's Flight Dynamics Facility (FDF) based on the most recent orbit determination solution. Tracking and Data

Relay Satellite (TDRS) ephemerides are also obtained through the FDF. Due to network security considerations, GlastCam cannot directly query the FDF network for the most recent spacecraft ephemeris. Instead, the Operations engineers need to manually obtain the files and place them in an ephemeris directory from which GlastCam can query them. During its start-up sequence, GlastCam goes through the ephemeris files and creates a linked list database based on their start and end times. This is necessary because the ephemeris files may overlap in time, or have gaps; rather than requesting that the FDF modify their process, we made GlastCam robust to its idiosyncrasies. During non-contact periods, GlastCam draws the GLAST ground track and TDRS locations by looking up the linked list database and interpolating the ephemerides based on two of the most recent data points. The linked list construct allows GlastCam to be insensitive to irregularities in FDF ephemeris deliveries.

Once the data are processed, all that remains is to display them. The enabling technology for this functionality is the OpenGL graphical library². OpenGL provides object libraries that can be linked with your application, and C bindings so you can call its functionality through C function calls. OpenGL is open source, well documented^{3,4}, and free. It is well supported on OSX and Linux. MS Windows ships with only an early version of OpenGL, but the open source community provides solutions for bringing it up to date⁵. Depending on your ambitions, the early version may well be all you need.

To maintain portability, OpenGL does not provide any capability for creating windows or handling user interfaces like keyboard and mouse events. Several good graphical user interface libraries exist. We used the freely-available GL Utility Toolkit (GLUT), in part because, like OpenGL, it also is portable across OSX, Linux, and MS Windows.

OpenGL and GLUT provide the infrastructure for drawing and managing a graphical display. It is up to the visualization developer to decide what to display. The following sections describe features we included in GlastCam and LROCam. In addition to increasing overall situational awareness, we included features to facilitate consistency checks of onboard measurements with each other and with independent ground-based sources.

A. GlastCam Features

GlastCam takes in the realtime telemetry stream and continually updates four distinct graphical displays that give immediate, at-a-glance insight into the spacecraft's situation. See Figure 2.

The first window, the “Cam” view, shows a 3D view of the spacecraft, viewable from any angle, with the Sun, Earth and Moon, and various informative telltales painted on the sky. The spacecraft's solar arrays and high-gain antenna are articulated based on gimbal telemetry. The star background uses only the GLAST star catalog. Instrument and star tracker fields of view are projected on the sky. Useful reference frames may be depicted by axes embedded in the spacecraft model, as well as by grids projected on the sky. The instrument fields of view, commanded inertial targets, as well as Sun- and Earth-related attitude constraints, are also projected on the sky. The Sun vector (measured and predicted), local magnetic field vector (measured and predicted), and system angular momentum vector are computed and depicted as vectors embedded in the spacecraft. Tracking and Data Relay System (TDRS) satellites and ground stations locations are computed and depicted (see Figure 3). Onboard estimates of TDRS locations are depicted, as well as the high-gain antenna beam and range of motion, providing easy checks on communication-related questions: Is the antenna pointing where flight software commands it? Is the command within the antenna's range of motion? Is the TDRS occulted by the Earth? Furthermore, the omni S-band communication system possesses interference regions, where coverage is weak. Depicting this interference region gave useful insight into the relationship between spacecraft attitude, ground station location, and the S-band communication link. When the spacecraft is out of contact, the Cam view does not attempt to propagate or predict the spacecraft attitude. Instead, the 3D display of the spacecraft is replaced by a “standing by” message to avoid misleading the casual observer (see Figure 4).

The second window shows a world map, displaying the positions of GLAST's orbit, ground stations, TDRS satellites, and several other features of special interest. This window does continue to display data when the spacecraft is out of contact, as these predictions are useful and easy enough to maintain. The position of the Sun, and its antipodal point, are important as they relate to GLAST's orbit, so they are both depicted. The map is divided into day and night as they are at ground level, but more important to us is the portion of the orbit eclipsed by Earth's shadow at GLAST altitude. A horizon “circle” shows the portion of the Earth visible from the spacecraft. The orbit trace has points drawn at intervals of ten minutes, making it easy to see, for instance, how much time until the next eclipse entry. The South Atlantic Anomaly is drawn on the map, since it is of importance to the GLAST instruments. The orbit track may be based on live spacecraft telemetry, or on externally provided ephemerides; the latter is especially needed when the spacecraft is not in contact. To make clear which source is being used, the orbit track is drawn as a dashed or solid line, respectively.

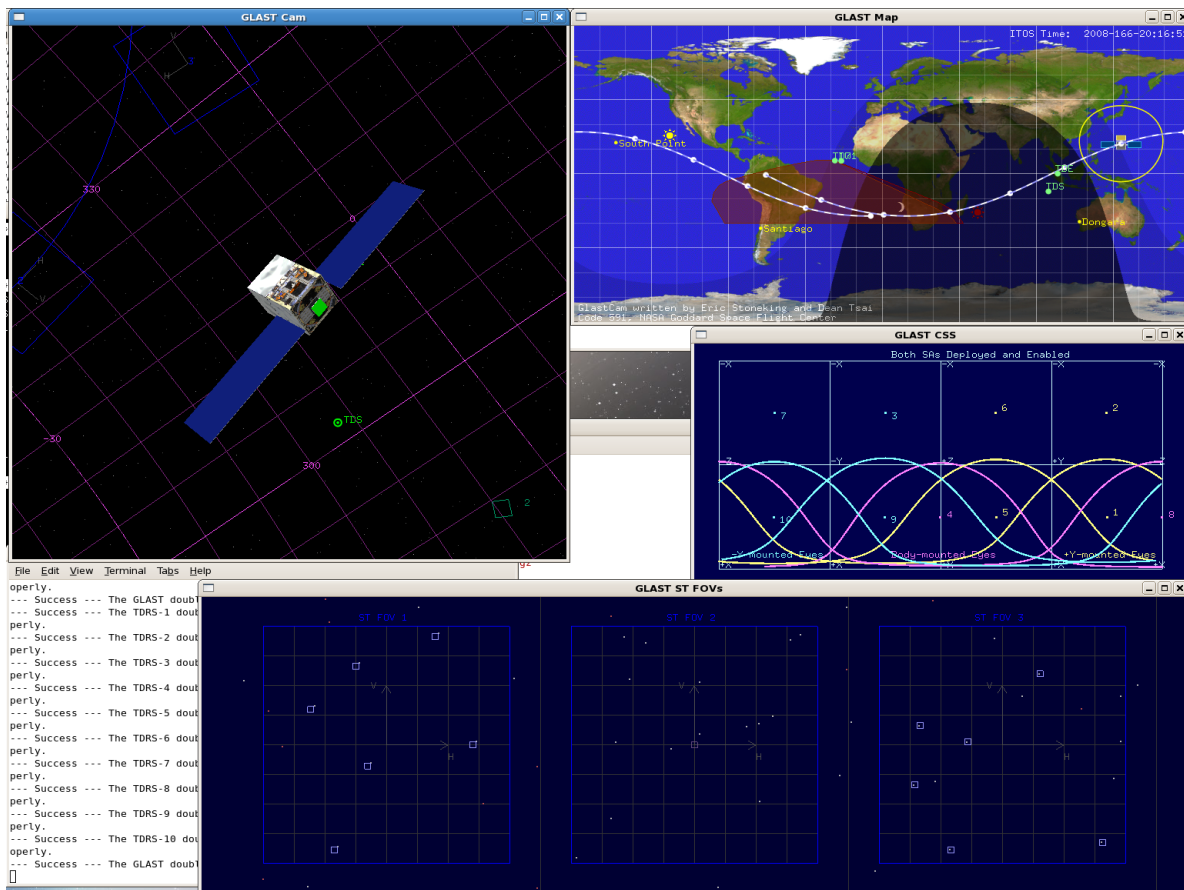


Figure 2: GlastCam's Default Four Windows Display

The remaining two windows provide more specialized insight into the functioning of the star trackers and coarse sun sensors. The star tracker window depicts the sky as seen through the star tracker fields of view. The stars are taken from the star catalog, and projected according to the spacecraft attitude estimate and star tracker mounting alignment. Star tracker tracking window locations and status are provided through telemetry. Star ID numbers may be optionally shown, providing a check that the star tracker has properly identified the stars it is tracking. The star tracker axes and some rule lines are also drawn.

The coarse sun sensor is the most specialized of the four GlastCam windows (see Figure 5). Consider a unit sphere centered on the spacecraft, then unwrapped and projected flat just like the Earth map in the Map window. Onto this flattened unit sphere, we project the boresight orientation of each coarse sun sensor, with an identifying tag. These boresight locations are derived from the known mounting of the sun sensors, and on the predicted or telemetered orientation of the solar arrays; eight of the ten coarse sun sensors are mounted on the solar arrays, the remaining two being fixed on the main spacecraft body. Based on the raw telemetry from each sensor, we draw a curve showing how far from its boresight each sensor believes the Sun to be. (On the unit sphere, these curves are small circles. On the planar projection, they are distorted.) Since there is only one Sun, all these curves should intersect at a single point, and that point is the direction of the Sun as seen from the spacecraft. This window provides consistency checks among the various sun sensors, as well as confirming the proper orientation of the solar arrays. If, for example, one solar array did not deploy properly, all the sun sensors mounted on it would yield information inconsistent with the assumption of a deployed array. This was used as an independent confirmation of solar array deployment.

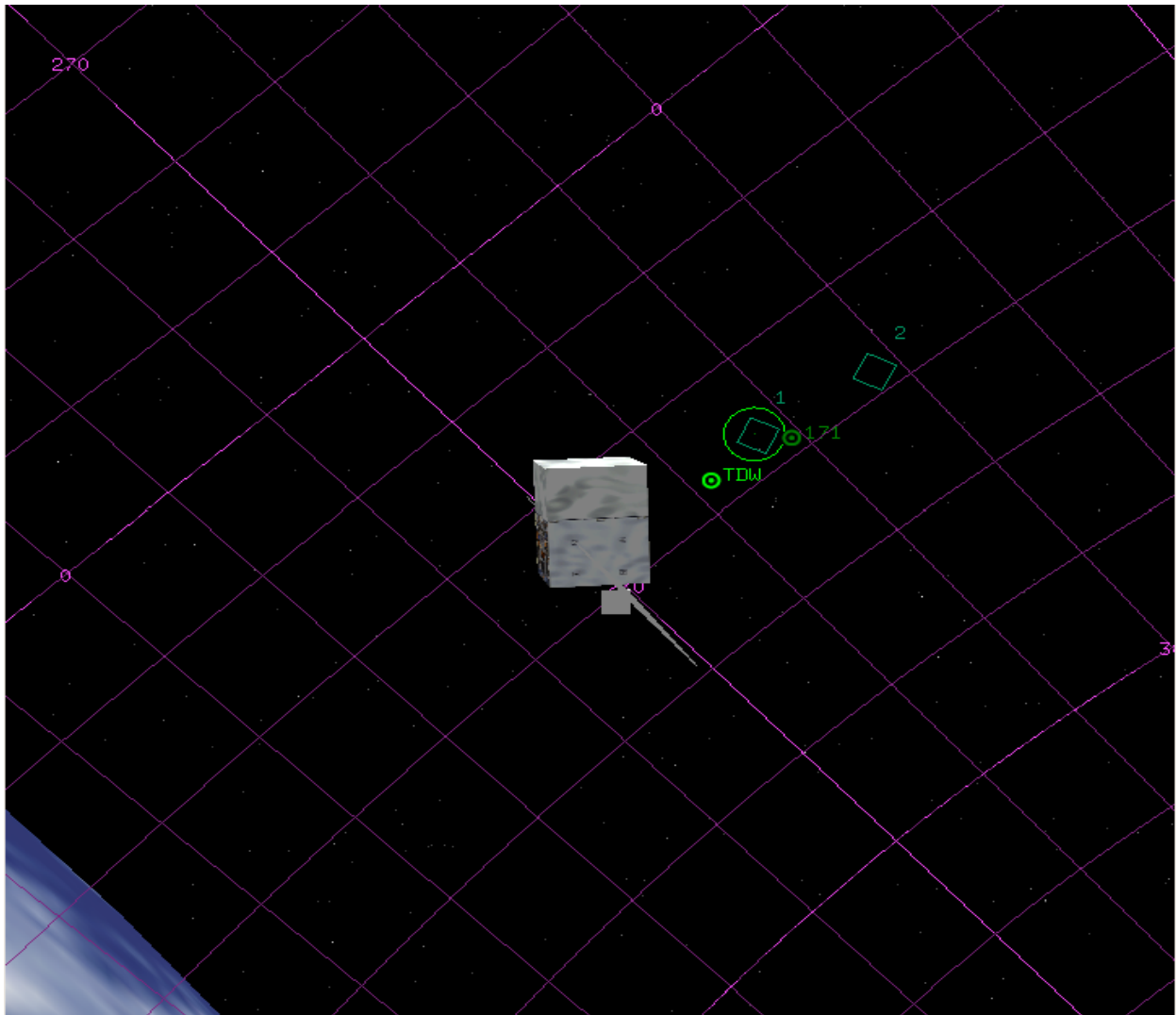


Figure 3: Ku Band Antennas FOV during Contact

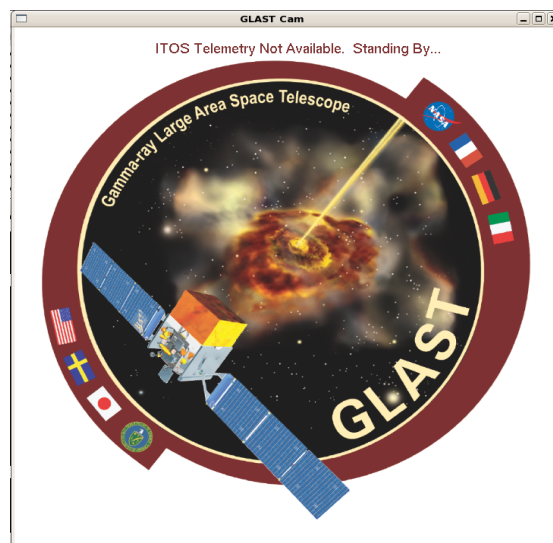


Figure 4: GlastCam's Standby Message

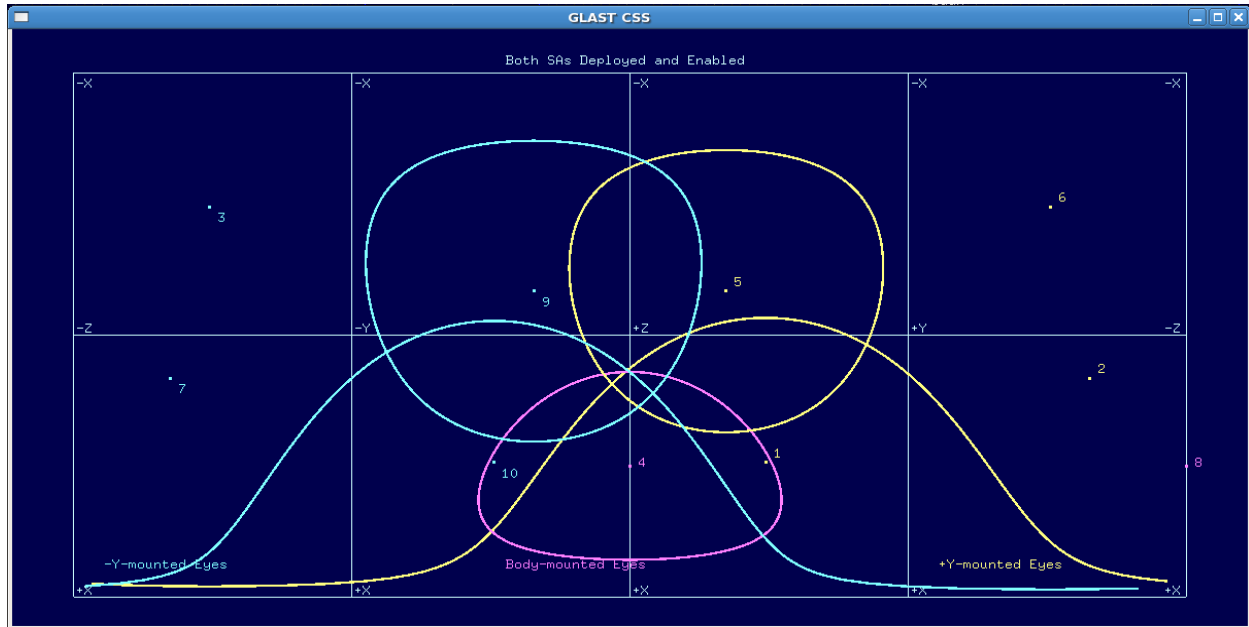


Figure 5: Coarse Sun Sensor Read-out and Sun Location in Body Frame

B. LROCam Features

Three windows, updated at the telemetry downlink rate, make up LROCam. A 3D Cam view is similar to that of GlastCam. An Earth Map is also displayed projecting LRO's orbit onto the surface, and a Moon Map displaying LRO's lunar-centric orbit is projected on the Lunar surface. Differences between LROCam and GlastCam's Cam and Map displays are due to the unique needs of each mission. One obvious example of this is that LRO is orbiting the Moon while GLAST is in orbit about Earth.

Although derived from GlastCam, LROCam is not identical in features. LROCam had the options to toggle on and off various sets of axes, depending on the mission phase. For example, during the commissioning phase, the attitude solution from one star tracker was used as the attitude feedback source. Therefore, this attitude was mapped to the spacecraft body and displayed on the Cam. However, during the nominal mission phase, the Kalman filter derived attitude estimate was being used as the attitude feedback and therefore, it was displayed. Another option on LROCam was the different antenna fields of view (FOVs), one of Ka-band communication and one for S-band communication. Each could be toggled on or off as desired.

LROCam's Moon Map is similar to the Earth Map, with the obvious distinction. Displaying LRO's orbit on the Earth Map provides critical visualization for the Earth-based communication link, but does not give insight into some naturally lunar-centric phenomena. For example, one cannot determine if LRO is at the lunar north pole descending towards the lunar south pole or the other way around from looking at the Earth Map; the Earth Map simply shows LRO's orbit as a small line. Looking at the Moon Map, one would be able to determine this easily. This feature of LROCam proved to be useful when LRO began nadir pointing operations with the solar array tracking.

III. On-Orbit Experiences

A. GLAST Experiences

GLAST was launched from Cape Canaveral on June 11, 2008. During its 60-day checkout period, GlastCam was in constant use, and was instrumental in two particular episodes. These both occurred during star tracker checkout; one highlights the value of planning ahead for contingencies, the other the value of being able to do things in real time.

Ground-based verification of star tracker polarity is notoriously tricky. One good way to do it is to use a specialized stimulator after the star trackers have been mounted on the spacecraft. On GLAST, we took another approach, using a combination of vendor documentation and night sky testing. We believed that we had it right, but



The next step in star tracker checkout was an intermediate-level precision adjustment of the inter-tracker alignment, a necessary preliminary step to facilitate the definitive arcsecond-level calibration. Watching GlastCam, it occurred to us that we could *see* the alignment error as a small offset between the predicted star location and its tracking box. Figure 7 shows that each tracking window has properly identified a catalog star (the blue and white numbers match), but each tracking window is slightly below and to the left of its star's predicted position. That common offset is due to the star tracker alignment errors that had not yet been calibrated. We had not anticipated a desire for adjusting star tracker alignments in GlastCam, but since we had full access to the source code, we had full capability to add such an adjustment very quickly. Adding some fine grid lines to the display, we introduced an

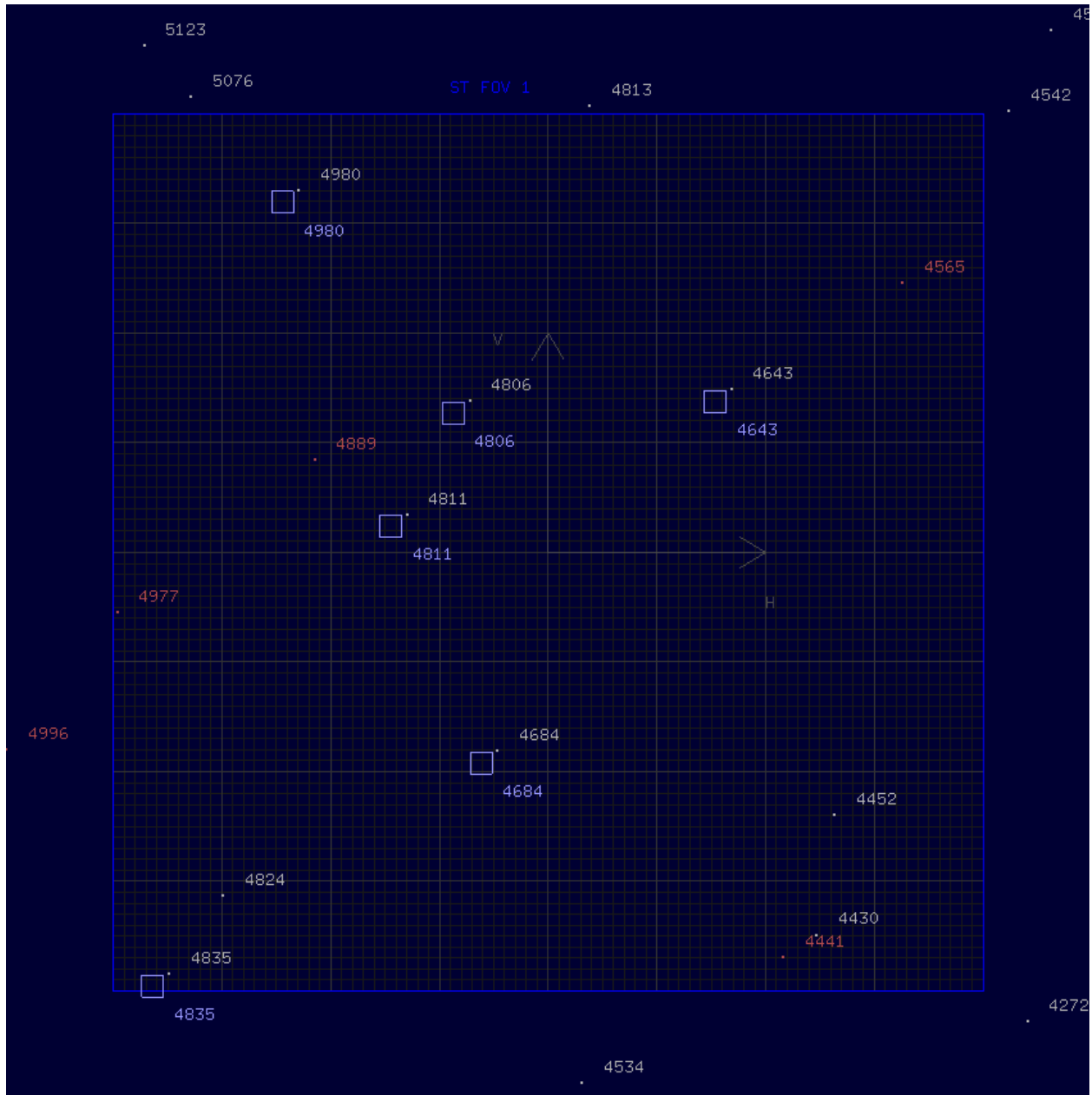


Figure 7: Star Tracker 1 Field of View Before Intermediate Alignment Adjustment

extra quaternion into the alignment chain, then adjusted it, re-compiled and ran GlastCam, and evaluated the adjustment. It took only a few iterations, taking only a few minutes, to put the stars solidly in their tracking boxes, yielding the parameter adjustments to upload to the spacecraft (see Figure 8).

Although GlastCam was written by GNC engineers primarily for our own specialized needs, the power of visualization proved widely applicable and very popular. Thermal engineers used it to check Earth illumination. Comm engineers used it to predict dropouts due to antenna interference regions. Everyone used it to answer basic questions that they might otherwise have had to bring to us. It was easy (and free!) to mount GlastCam on every workstation, easy to add features as they were requested, and easy for each user to configure their view to their particular needs. During the course of on-orbit checkout, we added numerous features at the request of other disciplines. Among these are the S-band antenna interference region, ground stations in the Cam view, fields of view of the Gamma ray Burst Monitor instrument, a gamma ray source catalog, the high-gain antenna range of

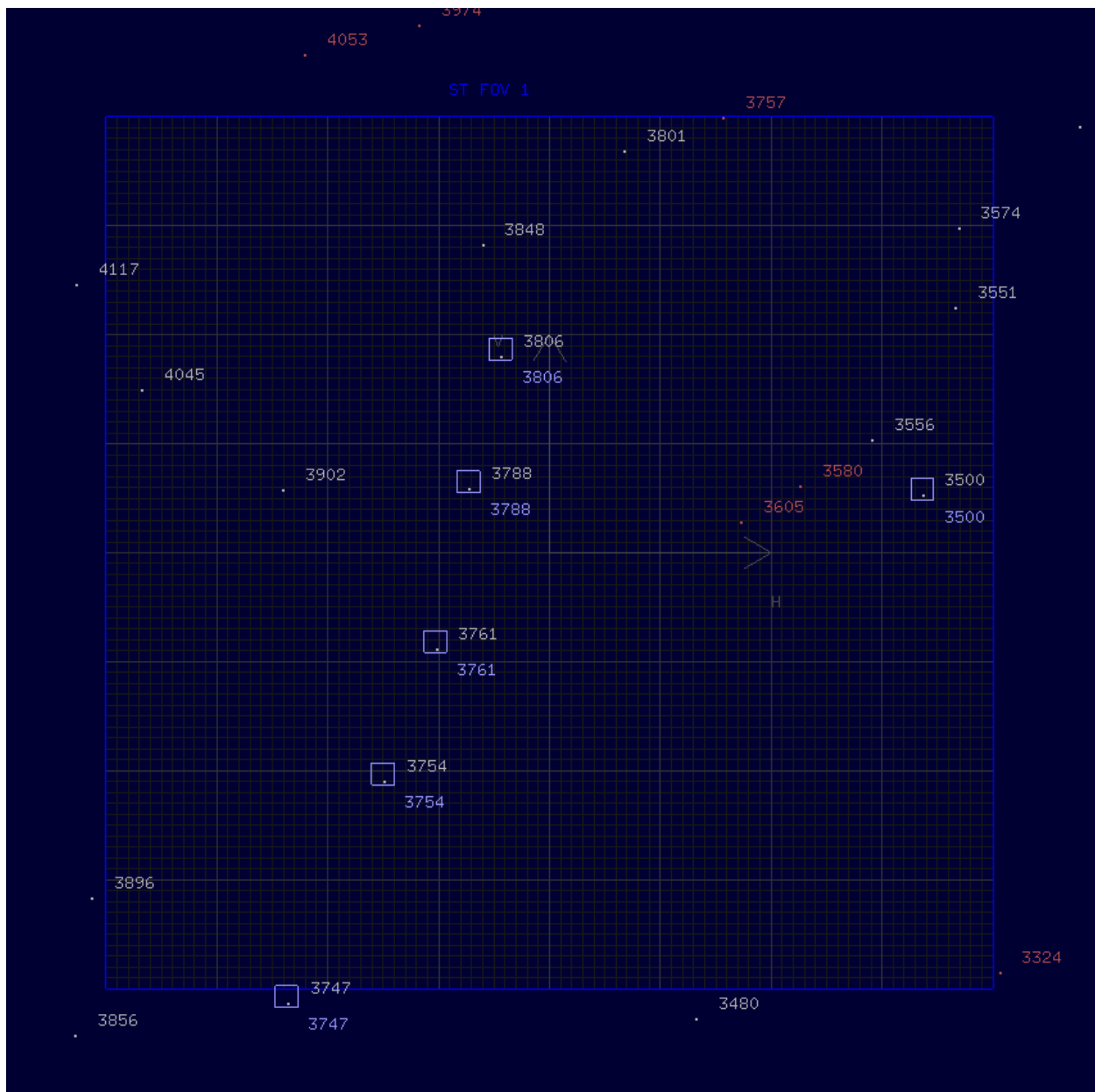


Figure 8: Star Tracker 1 Field of View After Intermediate Alignment Adjustment

motion, and the anti-Sun point on the Map view. Full access to all parts of GlastCam was the key to this adaptability and flexibility.

B. LRO Experiences

LRO launched on June 18, 2009 aboard an Atlas V launch vehicle from Cape Canaveral. LROCam was initially used only by the GNC Team. Within a week after launch, however, LROCam was deployed on all stations in the launch support room (LSR) as well as some key workstations in the mission operations center. During the early operations and commissioning phases of LRO's mission, the GNC team, as well as other subsystems and the systems engineering team, found LROCam very useful. Soon enough, LROCam became a favorite of everyone supporting LRO in the MOC and the LSR.

Upon separation from the launch vehicle, LRO is designed to autonomously place itself in a safe sun-pointing inertial orientation. LROCam's 3D window was useful in showing the slew to the sun during this exciting initial event (see Figure 9). In addition, the Earth Map gave an idea as to when the next ground station would occur. On occasion, when a communication link was not easily established, it was useful to look at the Earth Map to determine

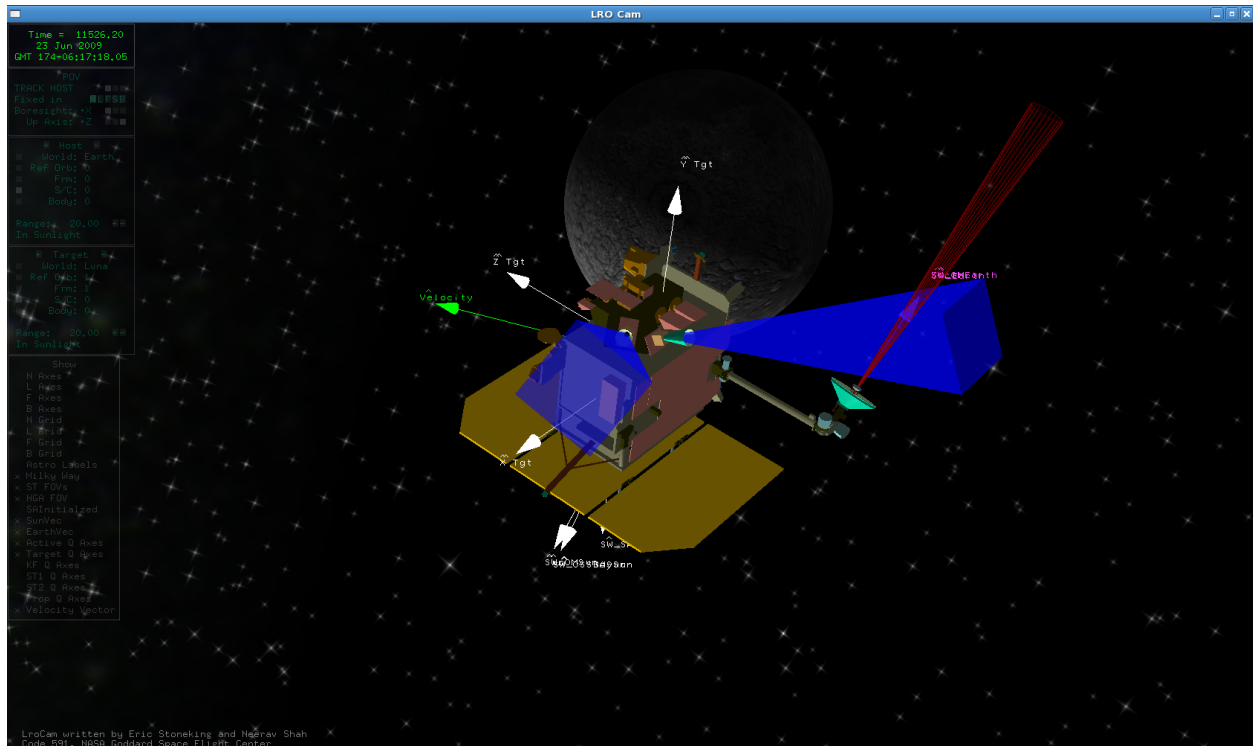


Figure 9: LROCam with Various Vectors and Fields of View

whether LRO was in the ground station field of view.

En-route to the Moon, the thermal subsystem found the temperature on the solar array gimbal drive to be too high. With the aid of LROCam, we were able to see that the Sun was directly on the gimbals. Therefore, an attitude offset was placed on the spacecraft to get the Sun off the normal to the spacecraft. We continued to fly LRO with a 30-degree offset so as to alleviate the heat build up on the solar array gimbals. LROCam helped in verifying the offset attitude was done correctly, and in the right direction.

During the lunar insertion maneuver sequence, LROCam was used in aiding to predict star tracker occultations by the Moon. After the first orbit insertion maneuver, LRO underwent an unplanned safe hold entry due to both star trackers being occulted. The details of this error can be found in Shah, et al⁶. LROCam was useful in providing corroborating evidence as to when the STs went into moon occultation, and aided engineers in determining the ultimate cause of the anomaly. The lunar orbit insertion (LOI) maneuvers were a series of thruster maneuvers the longest of which and mission critical was LOI-1. Without a successful LOI-1, LRO may not have had a successful mission. Therefore, it was important to view this sequence. So, during the maneuver, LROCam was running and capturing screenshots. The screenshots were made into a movie and distributed to project management and showed to colleagues. Displayed on the movie were the attitude controller errors, attitude axes and the Moon in the distance getting larger.

Quaternions have several properties that make them useful and popular for parameterization of spacecraft attitude. Ease of interpretation is *not* one of those properties. Even a seasoned practitioner may find it a challenge to answer simple questions like: Are these two quaternions close to each other? Is the difference mostly about one axis or another? Since LRO's attitude telemetry is displayed as a series of quaternions, visualizing these different quaternions was extremely useful, especially during a maneuver. LRO executes a maneuver by changing a target attitude and then selecting it as the source for the spacecraft to use, instantaneously causing a jump in the attitude error that the controller then reduces. With the option to toggle on and off various axes, selecting the current attitude and the target attitude allowed the spacecraft team to see what the spacecraft was actually doing, rather than decipher a quaternion.

The Moon Map proved useful once LRO was tracking a nadir attitude around the Moon. During this time, some confusion arose as to where LRO was flying. Telemetry from the solar arrays superficially suggested that the solar arrays were receiving no sunlight over a portion of the orbit, meaning that LRO was being eclipsed by the Moon. LROCam, in conjunction with other corroborating telemetry, was used to verify that LRO was in a full-sun orbit, and correlated the features seen in the power telemetry to comm dropouts and solar array rewind motion. This

example shows the power of combining the Cam view and the Map view, which integrate many items of telemetry, to give the situational awareness to properly interpret one confusing item of telemetry.

IV. Concluding Remarks

A word is in order regarding installation of GlastCam on operational workstations. For security and reliability, the GLAST Mission Operations Center places its workstations on a dedicated network, and discourages installation of any non-certified software. Before GlastCam could be installed in the MOC, it underwent thorough developmental testing at a separate development facility. Once checked out there, GlastCam was installed in the MOC, and underwent extensive “run in” during prelaunch operations simulations. By the launch date, we had developed the confidence to enable us to modify and recompile the code on the MOC workstation. GlastCam was classified as non-mission-critical since the traditional telemetry display pages were still available. Should GlastCam have failed in some way, the most severe remedy required would have been to switch to another workstation and fall back to display pages. Engineers would still be able to monitor the spacecraft the traditional way.

As Yogi Berra said, “You can see a lot just by looking.” Visualization is a powerful tool with many applications to spacecraft operations. Visualization doesn't replace traditional display pages completely, but greatly enhances situational awareness and facilitates rapid debugging and recovery processes. In the intense atmosphere of early mission operations, both of these are highly valuable. Freely available resources like OpenGL bring the means to develop visualization tools within the reach of a small team of engineers, who may know more about their spacecraft than about software development. The resulting product has the advantages of customization, adaptability and transparency. Both the GLAST and LRO experiences attest to the power of visualization, and to the advantages of having full access to double-check and modify the visualization tool to adapt to the needs of the operations environment.

References

¹STK, Satellite Tool Kit, Software Package, Analytical Graphics, Inc. URL: <http://www.agi.com/>.

²OpenGL, Software Package. URL: <http://www.opengl.org/>.

³OpenGL Architecture Review Board. *OpenGL Programming Guide*, Third Edition. Addison-Wesley, Reading, Massachusetts, 1999.

⁴Richard S. Wright, Jr., Benjamin Lipchak, Nicholas Haemel. *OpenGL SuperBible*, Fourth Edition. Addison-Wesley, Upper Saddle River, New Jersey, 2007.

⁵GLEE, GL Easy Extension Library, Software Package. URL: <http://www.opengl.org/sdk/libs/GLee/>

⁶Shah, N., et.al., "Launch and Commissioning of the Lunar Reconnaissance Orbiter (LRO)," AAS 10-085, American Astronautical Society (AAS) Guidance and Control (G&C) Conference, Breckenridge, CO., February 2010.