NASA/TM-20230018222

Improvements to GCAS Visualization Functionality

Vaughn M. Richard and Bryan W. Welch
Glenn Research Center, Cleveland, Ohio

March 2024

NASA STI Program Report Series

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION.
  Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM.
  Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT.
  Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONTRACTOR REPORT.
  Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION.
  Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION.
  Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION.
  English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

NASA/TM-20230018222

Improvements to GCAS Visualization Functionality

Vaughn M. Richard and Bryan W. Welch
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

March 2024

Acknowledgments

The primary author would like to thank his mentor, Bryan Welch, for his guidance and assistance throughout the project, fellow interns Joseph Symons and Sam Bloomingdale for their assistance in development, and Alia Smith and Matteo Restuccia for their editorial contributions.

This report contains preliminary findings, subject to revision as analysis proceeds.

Level of Review: This material has been technically reviewed by technical management.

This report is available in electronic form at https://www.sti.nasa.gov/ and https://ntrs.nasa.gov/

Improvements to GCAS Visualization Functionality

Vaughn M. Richard* and Bryan W. Welch
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Abstract

Analysis of communication systems is crucial to NASA objectives, whether located here on Earth or on a cosmic scale. The Glenn Research Center Communication Analysis Suite (GCAS) is software designed for analysis of communication systems throughout the solar system. This suite includes web-based visualization software that is used to simulate the data generated from the GCAS MATLAB® (The MathWorks, Inc.) capability. This report focuses on recent additions to the visualization portion of GCAS, including the motivation and methodology behind them. Notable additions include improved camera controls, such as novel first- and third-person settings, methods for displaying pertinent analysis information, finer control over timing mechanisms, and automated features designed to improve efficiency and memory usage.

Nomenclature

API	application programmable interface
FSA	file system access
GUI	graphical user interface
HTTP	hypertext transfer protocol
JSON	JavaScript object notation
NVF	node-focused view
VIS	GCAS Visualization Software

Symbols

θ	angle about the z-axis
Φ	angle off the x-y plane

1.0 Introduction

The Glenn Research Center Communication Analysis Suite (GCAS) is a software suite used to perform technical analysis on communication systems designed for use throughout the solar system. It performs the analysis in MATLAB®, a proprietary language developed by The MathWorks, Inc. (Ref. 1). From the analysis, a JavaScript Object Notation (JSON) file is constructed with the position and velocity state information of the relevant objects (planets, satellites, ground stations, etc.) (Ref. 2). This data can then be used in the visualization aspect of GCAS. The visualization software (VIS) is used to simulate the GCAS analysis. The visualization tool is used to represent the scenario analyzed within GCAS using a web browser. Although it works best with the Microsoft Edge® web browser, it also works with the Google Chrome® browser. Figure 1 is an example of a scene generated by VIS.

*NASA Office of STEM Engagement (OSTEM) Spring 2023 intern, University of Chicago undergraduate.

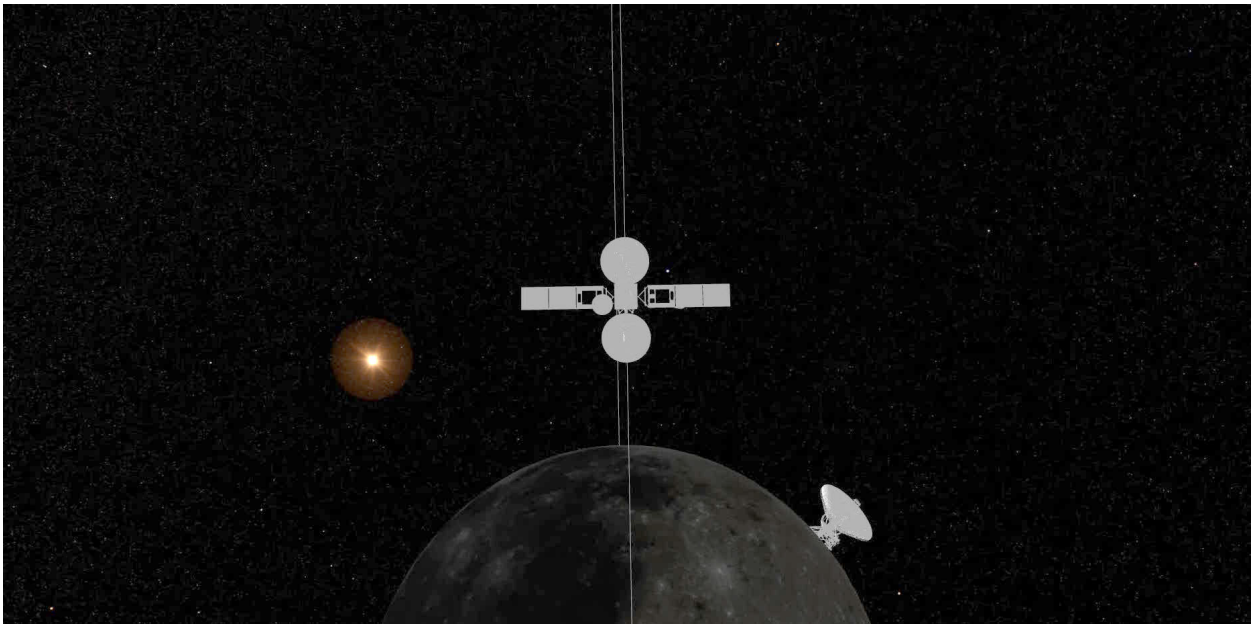

Figure 1.—Example of VIS-generated scene.

The visualization animates the scenario through the data contained at the various timesteps, where the timestep is a discrete unit of time that is configured from the GCAS analysis capability. A timestep is the smallest time increment available in the visualization, configurable in the MATLAB® scripts. Each timestep contains the positional data for all objects in the scene at a real-world time. The timestep will automatically progress but may also be controlled by the user, as described in the following paragraphs.

Three main mechanisms support control of the visualization. The first is the bottom toolbar, which includes buttons to pause, to step through the visualization, and to skip to the beginning and end. The toolbar also contains a slider bar that allows a user to drag the visualization to a desired time. On the right, a number of buttons can be found that are useful for analysis, such as screenshot and video recording buttons.xxx

The second mode of integration concerns the visualization itself. By clicking and dragging, the user can interactively control the image on which the visualization is focused. In many views, it is also possible to scroll with the mouse wheel to change the distance from the object being observed.

The third and most varied mechanism of control is a controls graphical user interface (GUI), which can be found at the top right of the visualization. This GUI contains options to change camera settings, lighting parameters, and object representation, among others. Most entities loaded into the scene can be controlled from this GUI (Figure 2).

Further information about the software can be found in earlier NASA reports (Refs. 3 and 4).

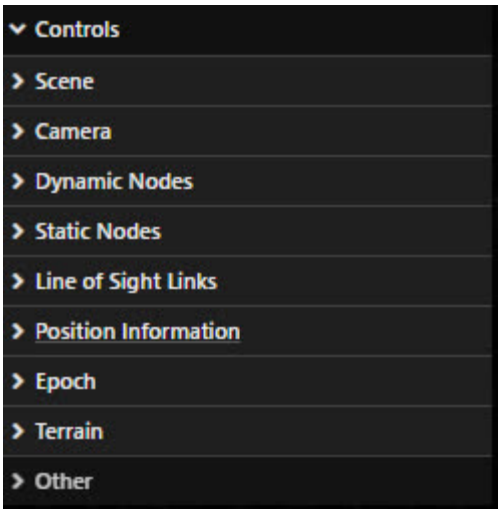

Figure 2.—GUI control options.

2.0 Camera Controls

The ability to control the view effectively is crucial to the functionality of the visualization tool. Earlier iterations of VIS had relatively few camera options, such as changing the camera field of view and changing the target object of the visualization, among other more rudimentary control mechanisms. The software lacked fine control when viewing the scene, especially when looking at nodes, which are defined in Section 2.1. Several improvements were developed for this purpose. The most significant additions were those for node-specific views. The default controls were also improved and a new set of controls that allowed for free movement of the camera was added.

2.1 Node-Relative Controls

VIS was developed to help users understand how nonplanetary objects (here called nodes) behave in a GCAS-generated scenario. Common objects generated include satellites, airplanes, ground stations, and rovers.

Without proper control over how a node is viewed, it can be difficult to understand its behavior, even within the visualization. Further, each node type has drastically different movement patterns, increasing the complexity of designing intuitive controls. The views described in Sections 2.1.1, 2.1.2, and 2.1.3 were developed to enhance a user's ability to manipulate the camera to desired positions in relation to nodes.

2.1.1 Node-Focused View (NFV)

This view is available for any node. When analyzing a node, the ability to observe the scene from its perspective is useful for understanding its behavior. The most powerful system in VIS for this is the NFV. This view generates the view of a camera looking towards an object from the node's perspective. One use case is monitoring how a satellite views a specific ground station. When selected, the camera is placed on a tangent between the node and a second chosen position, looking towards this position.

The NFV requires the node's position and a second focused position. Any object in the scene, the primary body's horizon, or the Sun's position may be selected as this second object. A drop-down menu for selecting this second position is called the Object Focus Option (Figure 3). The initial camera position

is found by taking the vector pointing towards the focused position from the node's position, normalizing and negating it, and then setting the camera position to the node's position plus this vector.

Two values must be set properly to ensure a useful view. The most straightforward value is the camera's look-at position, which simply faces the camera toward this set position. Notably, changing solely the look-at position will often produce poor visualizations because the camera's orientation will not be well defined. This means that although the object will be in view, it could appear upside down or offset from the desired view by any degree.

To rectify this issue, the camera has a value called the up vector that must be set prior to setting the look-at position. Setting the up vector will ensure the top of the camera's view is in the desired orientation. Figure 4 provides a visualization of this concept. As an illustration, if a camera looking at Earth had the North Pole as its up vector, the standard world view would appear. If, however, the South Pole was the up vector, Antarctica would appear at the top of the image generated for a camera in the same position.

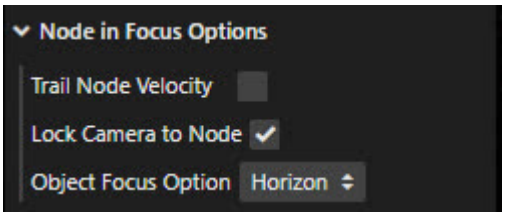

Figure 3.—Node-focused GUI options.

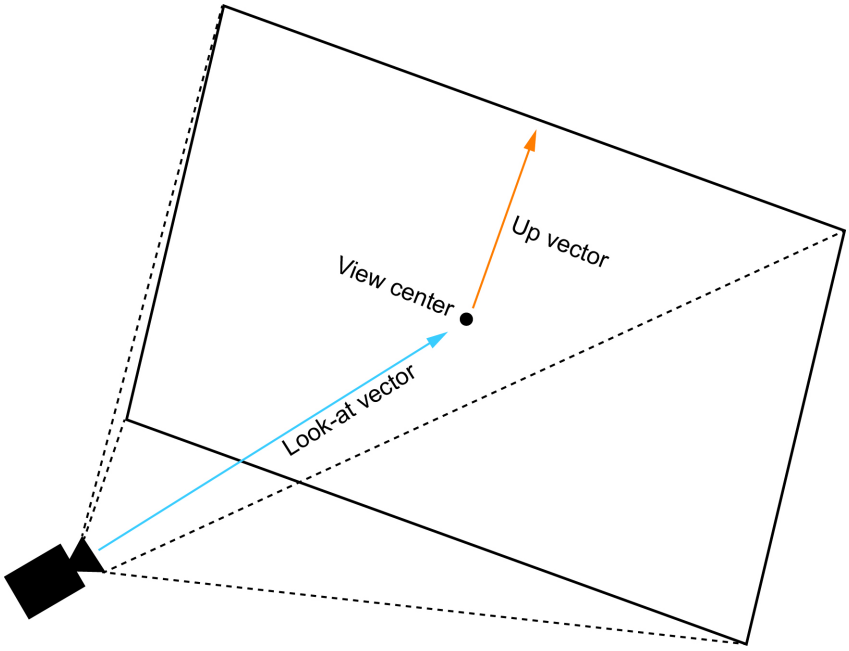

Figure 4.—Depiction of camera orientation.

For views close to the primary body, the best view is one in which the up vector is the normal to the body. The normal to a body is the vector perpendicular to its surface. For a human standing on a spherical planet, this vector would point from the person's feet to his or her head. Because of this, it is also the natural human perspective. For people standing on Earth, their up direction points to the sky, away from the ground.

One useful view to have is one that looks towards the horizon. Because this point does not exist naturally, one must be created. For a node at the position (a, b, c), the look-at position should be

$$\text{If } b = 0 \text{ then} (a,0,c) + (0,1,0)$$
$$\text{Else } (a,b,c) + \left(1, -\frac{a}{b}, 0\right)$$

The rationale behind these vectors is in the appendix. For each view, the up vector should be set to the normal, which equals *norm*(a, b, c).

The camera may also be rotated to different angles. Due to the camera position being from a third-person perspective, any rotations must be about the node. For the VIS implementation, a rotate about function native to the JavaScript visualization package was used (Ref. 5). This rotates a point around a given axis, providing the desired functionality with axes relative to the chosen node. This may be replicated by mirroring this functionality or adjusting the Φ and θ angles in the user's own system. Figure 5 provides a visual representation of these concepts. It is crucial to position the camera after these changes have been made; otherwise, these rotations may shift the location the camera looks at. Rotation can be controlled using an intuitive click-and-drag method, much like the controls found in design software.

Finally, the distance from the object may be changed by scrolling with the mouse wheel. This simply scales the camera's distance to the node.

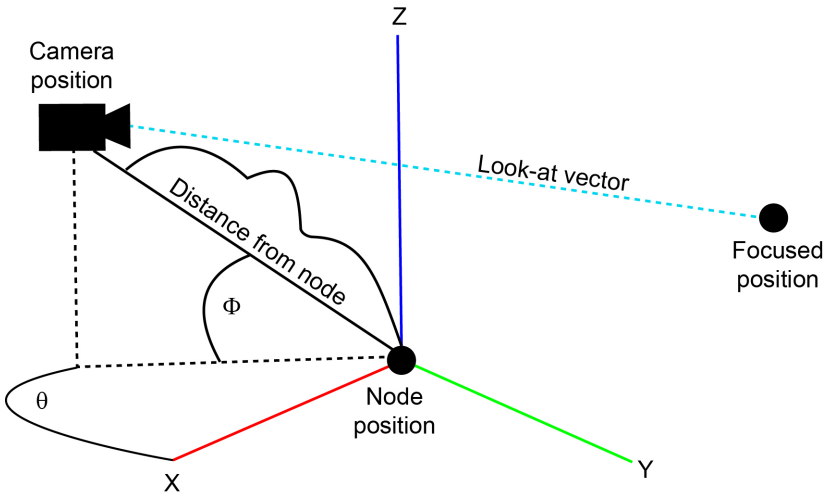

Figure 5.—Node-focused view.

2.1.2 Trail Node Velocity View

This view is available only for objects with velocity, that is, not with ground stations. A natural position for observing a moving object is one relative to its velocity. For example, when observing the behavior of a satellite or airplane, it is desirable to have the camera locked in a position that moves with the object, that is, a camera position that moves with the node velocity. This was the motivation behind developing this camera view. It has the same rotation mechanisms as the NFV, but the up vector is the normal of the object to its primary body and the initial camera position is the normalized and negated velocity vector. It always looks towards the selected node. Otherwise, it is analogous to the NFV.

2.1.3 Site Analysis View

The site analysis view is only available for stationary nodes. Unlike the previous views, this one is from a first-person perspective. Its position is closely related to the surface node, but there is a height slider that allows the position of the node to vary from 0 m (at the node) to 100 m above the node, along the normal vector of the body.

This view shares the drop-down functionality with NFV, allowing the user to observe any of the objects from a first-person perspective at a surface. This allows tracking object visibility from such sites. The implementation here is much simpler; all that is required is a look at the operation.

Due to the difference in perspective, the site analysis view also has a different rotation mechanism. The site analysis view is a visual perspective that only applies to stationary nodes on the surface of a planet/moon; the view the user is attempting to achieve is called a local topocentric coordinate frame perspective, in which the local surface tangent plane is the horizontal view on the screen, and the up vector off that tangent plane is the vertical view on the screen. The easiest way of achieving this is through local rotations (Ref. 6). Given that this is a view close to the body surface, it is most natural to have the up vector set to the body normal.

The scroll wheel is used to zoom closer to or further from a node or planet/moon, but in this view, the camera is at the node's location. The rotation mechanisms described in this subsection are displayed in Figure 6.

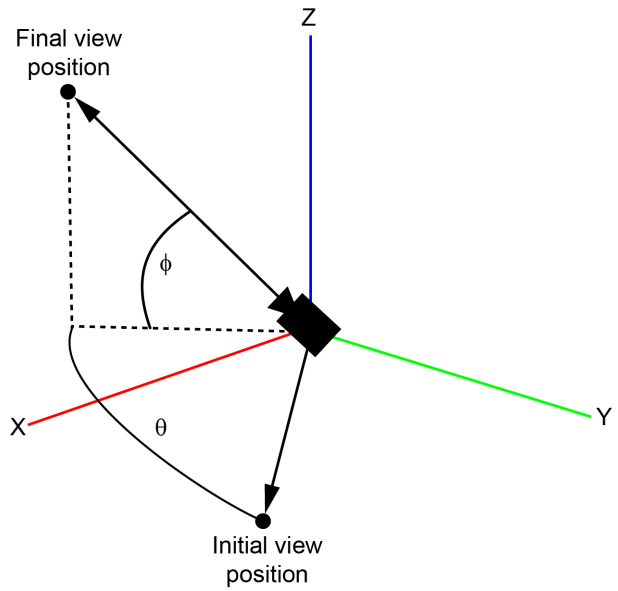

Figure 6.—First-person view.

2.2 Trackball Controls

The previous iteration of VIS used orbit controls for the default viewing. Orbit controls work by defining a constant up vector. The top of the camera always points towards this vector, limiting users to the domain of [–90, 90] on the Φ angle (refer to Figure 5 for Φ). This lends itself as a natural way to view planets because it is often useful to set the camera relative to an upward North Pole.

Although this can be a useful implementation, it was found to be limiting for VIS purposes; the gimbal lock along the Φ degree made it difficult to achieve certain desired views on visualization objects (Ref. 7). The solution to this was migrating to trackball controls (Ref. 8). These controls do not define such an up vector; instead, translating the camera is based on the camera's view of the object. This allows the freedom to reach any view of an object, but it comes with the tradeoff of being harder to control.

2.3 Free Controls

This version of VIS introduced renderable lunar terrain. The ability to navigate this is a useful function, so the free control mode was added to the visualization capabilities. This mode allows the user to traverse the scene on the surface of the body/moon, like in a video game. It is well suited for rough lunar terrain, allowing users to understand the scope of the terrain they want to analyze. These controls do not have a specific object in focus, but instead, allow the user to move the camera from a first-person point of view. The W, A, S, and D keys can be used to move forward, left, backward, and right, respectively. The shift and space keys can be used to reduce and increase elevation, respectively. A slider in the controls GUI allows for speed adjustments.

One important consideration when implementing free controls is how the view changes over curved surfaces. Previously, there was a problem associated with moving left and right that made the camera appear rotated with respect to the planet. Although this may be desirable for some implementations, such as free controls far off in space, it was a problem for these purposes. VIS's free controls were intended to be used near planetary bodies, so a view that people would have if they were at that position was desirable. This was rectified by setting the camera's up position to the normal of the body whenever the position updates.

3.0 Information Display

Visualizing a scenario helps in understanding spatial and operational behavior. As such, live displays of such information were added to improve scenario visualization. Both the position and link displays can be toggled by a selection button in the controls GUI. When enabled, displays are overlaid on the visualization that displays the selected information.

3.1 Position Display

The position display (Figure 7) supports showing the position of any object in the scene. The position displayed is relative to the current primary body, with units in kilometers. Additions to the main panel include a drop-down menu that allows for any object to be selected, as well as options likely to be used frequently (Figure 8). Once an object is selected, an option to remove it is added to the position display folder.

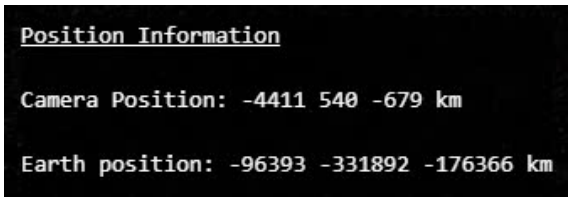

Figure 7.—Position display.

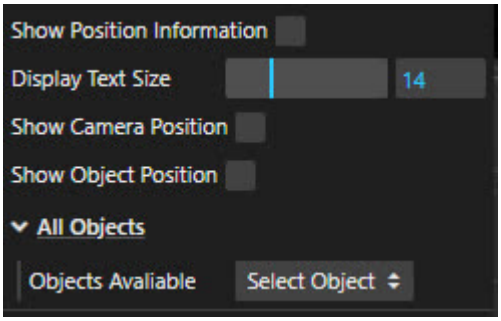

Figure 8.—Position information GUI.

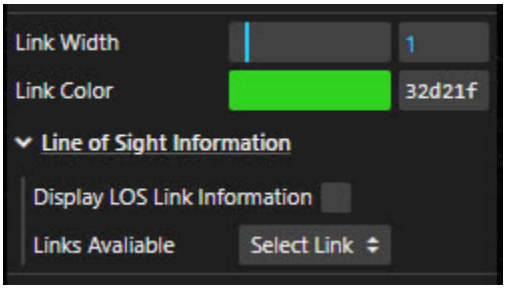

Figure 9.—LOS link GUI.

3.2 Link Display

Display link information follows rules similar to those that apply to position display. Both Dynamic Links and Line of Sight (LOS) links may be displayed, both having the same controls. Only two options exist for displaying link information: a toggle for displaying the link information that exists and a drop-down box for selecting a link of interest (Figure 9). An option makes it possible to remove the position information described in the previous section so that only the Dynamic Links and LOS links are displayed.

Once selected, the link information will display updates with every timestep. LOS links and Dynamic Links are two separate analysis functions. LOS returns if the two nodes can observe each other geometrically, while Dynamic Link will display one of several terms that GCAS can solve for (link margin, potential data rate, and required power), which typically change based on the varying relative motion between the nodes.

4.0 Time Control

While working through the visualization, users can find a specific interval (or intervals) of interest. Epochs were added to allow users to restrict the time domain to desired blocks.

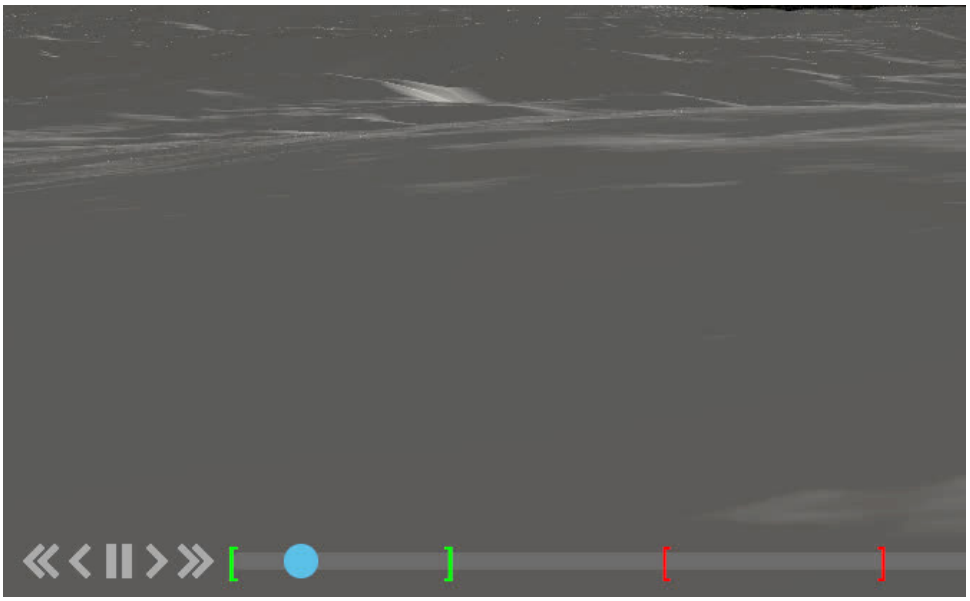

Figure 10.—Epoch example.

When it is enabled, the slider becomes locked to values within the domain of the selected Epochs. Epochs consist of a beginning and end, displayed as brackets ([and]). Figure 10 shows an example of two Epochs placed into a scene.

Epochs can occur at any time within the scenario. As usual, the user can drag the slider to explore a single Epoch; jumping between Epochs is as simple as dragging the cursor over them; alternatively, the buttons used for skipping to the beginning and end of the visualization in the standard mode can be used to toggle between Epochs. The Epoch brackets may be dragged to the desired positions, and Epochs may wrap around the end of the visualization time domain. Further functionality is described in the following paragraphs. This new feature was split up between control of all the Epochs or control of an individual Epoch. An Epoch tab was added to the controls GUI, which includes three buttons: Toggle Epoch mode, Lock to Epoch, and Add Epoch.

Toggle Epoch mode was added as a quick way to switch between standard functionality and one in which the time domain is locked to the Epochs. Even if Epochs are added, they will be ignored until this value is enabled. In other words, even when Epochs are configured, but scene animation is not limited to that Epoch until the user checks that mode. This mode was added to allow switching easily between normal usage and focusing on specific time periods of interest.

The Lock to Epoch function was added for scenes containing multiple Epochs, but when using this function, the user wishes to observe just one. While within the chosen Epoch, the user can turn on this function to limit the time domain to the current Epoch. This ease-of-use addition allows users to focus on a desired Epoch without the need to remove others of potential interest from the scene.

The Add Epoch button allows adding new Epochs to the slider by placing them a manageable distance from the previous Epoch. Epochs must be kept long enough to be relevant and far enough apart chronologically to remain independent. If the Epochs are too close to each other, this function will simply place the new one at the end of the visualization.

Per-Epoch controls can be accessed by interacting with the Epochs themselves. For example, changing the position of an Epoch bracket is as easy as dragging and dropping it. Selecting an Epoch will open a panel that provides further control. On the ends of this panel, < and > buttons let users change an

Epoch's position by one timestep. The panel also has a trash can button to destroy the Epoch and a color button that allows the user to select a color for the Epoch. The latter buttons affect the entire Epoch (both buttons) while the prior affects only the bracket selected.

Scheduling is another detail to consider when implementing the Epoch capability. This VIS implementation uses the ordering of the Epochs for this purpose, updating the order on deletion and addition. An alternative approach would be to schedule the Epochs by the order they appear on the slider bar. This ensures that the time will always progress predictably, but it also reduces the customizability for the user. With the VIS implementation, the scheduling may be in any desired order. For example, if the entire scene includes Epochs A, B, C, and D in that order, the user can also instruct the visualization to play them in any other sequence desired, such as C, B, A, and D.

5.0 Automation and Data Management

The previous version of VIS had the ability to record videos, but due to limitations with the application programming interface (API) used to take them, they could only be downloaded in a WebM video format (Ref. 9). This file format is less than ideal for this purpose because it is incompatible with many popular applications, including the Microsoft Office® Suite. In contrast, an MP4 file does not have these limitations so it is typically preferred (Ref. 10). In the past, GCAS was packaged with a WebM-to-MP4 converter to allow users to work around this difficulty. In the new version of VIS, which uses Node.js server-side operations, the software now outputs an mp4 video to a desired location with no additional user steps required.

Automation also allows for more efficient data usage. VIS's web-based approach introduces stringent memory limitations, which means the browser's memory limit can be reached before the computer's capabilities are exceeded. This makes it essential to keep as much data off the browser as possible.

VIS encountered this error when a feature to take screenshots over a time domain (either Epoch-selected time zones or the entire scene time) was introduced. This feature took screenshots automatically; without proper data management, this could have required storing thousands of screenshots temporarily. Given that each screenshot could potentially be megabytes in size, the gigabytes of required storage would easily exceed the browser's capabilities.

The File System Access (FSA) API helps users overcome this issue (Ref. 11). This API cannot only create files on the user's computer but allows piping data directly to these files, eliminating virtually all memory strain due to the amount of data generated (Refs. 12 and 13). Figure 11 shows how data was processed in this scheme.

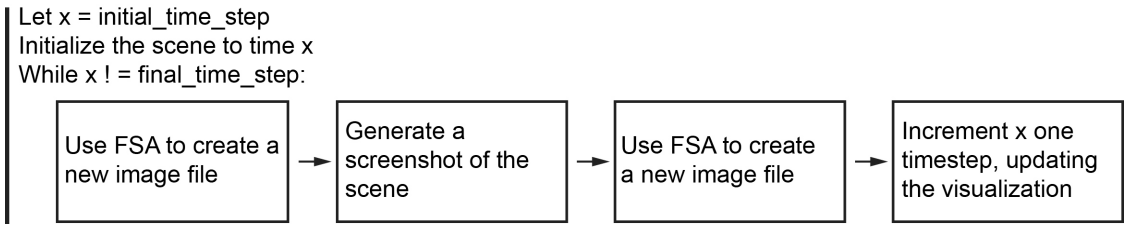

Figure 11.—Data processing scheme.

A similar piping approach could be applied to video recordings or any data-intensive function that requires server-side processing. This requires sending “blobs” or sets of data directly to the server side for storage as they become available (Ref. 14). This is a more traditional server-client approach and requires Hypertext Transfer Protocol (HTTP) data transfers (Ref. 15). Once all the data has been sent and processed, this data may be sent directly to the user, maintaining low browser memory utilization.

6.0 Conclusions

The ability to understand communication systems accurately and intuitively is crucial to achieving NASA’s objectives. The newest Glenn Research Center Communication Analysis Suite (GCAS) features were added to enhance control and ease of use to help project teams perform their analyses. They also addressed limitations in the previous version of the visualization software. Several more features, including an orientation display, additional node views, and improved parameter customizability, would further support the goal. Additional novel features will be desirable as new projects are begun or current ones progress.

Appendix—Horizon Vector Proof

Some interesting math is used to generate the visualization software's horizon option. Figure 12 offers a visual guide to the following explanation. The goal of this view is to orient the camera parallel to the surface of the body, that is, if the node's position was scaled to the surface, it would look at the horizon. This presents a challenge as there is no defined reference point for setting this view.

When considering a point on the surface, a simple way presents itself using some geometry and linear algebra. If a latitudinal line is found at the point, the tangent of this circle provides a vector that can be used to generate a desired look-at position.

Define the center of the body to be the origin (0, 0, 0). Now, let the point on the surface be defined as $p = (a, b, c)$, where a, b, and c correspond to the x, y, and z components, respectively. Also, assume the planetary object to approximate a spherical shape with radius r. With these assumptions, we know that

$$a^2 + b^2 + c^2 = r^2 \tag{1}$$

This can be rearranged to find the latitudinal line at position c on the z-axis:

$$a^2 + b^2 = r^2 - c^2 \tag{2}$$

With this equation in hand, the tangent to this circle may be found:

$$ax + by = r^2 - c^2 \Rightarrow y = -\frac{a}{b}x + r^2 - c^2 \tag{3}$$

This makes the resulting tangent vector a simple and easily computable result. The vector is of the equation $tangent = \left(1, -\frac{a}{b}, 0\right)$. The constant $r^2 - c^2$ term may be ignored as only the slope of the line is desired. If b is 0 (the point lies on the x-axis), then set the vector to be $\left(0, -\frac{a}{|a|}, 0\right)$. To set the camera's look-at position, the up vector must be set to the normal of the planetary body. Then, the look-at position may be calculated as follows:

$$\text{Up} = norm(a, b, c) \tag{4}$$

$$\text{Look-at} = (a, b, c) + \left(1, -\frac{a}{b}, 0\right) \text{ or } (a, b, c) + \left(1, -\frac{a}{|a|}, 0\right) \tag{5}$$

Even as the node moves away from the surface of the body, the vector in Equation (3) is still the desired direction to look toward. It provides a consistent view relative to the planetary body for a node. This means that along a radial line from the center of the planet through a position on the surface, every point should have the same horizontal view. Fortunately, doing this calculation at any point on such a line will yield the same horizon vector, as shown in the following equations.

Let $pos = (s, t, u)$ be the position along a radial line l from the center of the planet $(0, 0, 0)$. Let $pos' = \alpha * pos = (\alpha s, \alpha t, \alpha u)$ be another point along this expansion where $\alpha > 0$. For simplicity, let $\alpha s = s'$, $\alpha t = t'$, and $\alpha u = u'$. Consider the sphere each of these points lies on, much as was done for the point on the surface previously.

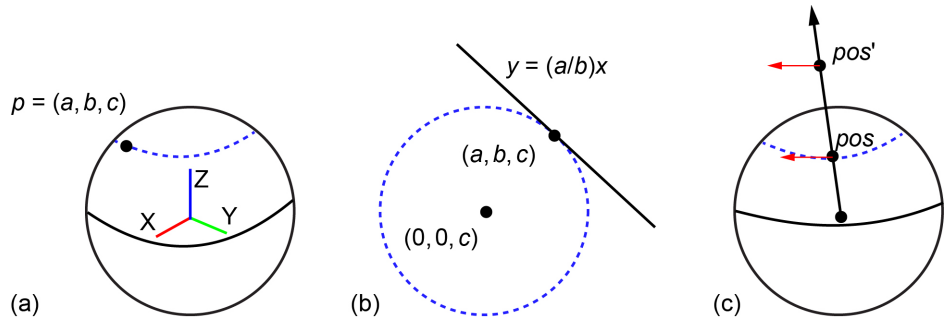

Figure 12.—Horizon view.

First, find the tangent line for *pos*. Let $\sqrt{s^2+t^2+u^2}=r$:

$$s^2+t^2+u^2=r^2 \tag{6}$$

$$s^2+t^2=r^2-u^2 \tag{7}$$

$$sx+ty=r^2-u^2 \Rightarrow y=-\frac{s}{t}x+(r^2-u^2) \tag{8}$$

Thus, the tangent vector for *pos* is (1, –*s*/*t*, 0). Now, find the tangent line for *pos*′

$$pos'=\alpha * pos \Rightarrow \sqrt{s'^2+t'^2+u'^2}=\alpha r: \tag{9}$$

$$s'^2+t'^2+u'^2=\alpha^2 r^2 \tag{10}$$

$$s'^2+t'^2=\alpha^2 r^2-u'^2 \tag{11}$$

$$s'x+t'y=\alpha^2 r^2-u'^2 \Rightarrow y=-\frac{s'}{t'}x+(\alpha^2 r^2-u'^2) \tag{12}$$

$$y=-\frac{as}{at}x+\alpha^2(r^2-u^2) \Rightarrow y=-\frac{s}{t}x+(\alpha^2 r^2-u'^2) \tag{13}$$

That means that its tangent vector is also (1, –*s*/*t*, 0). Because *pos* was an arbitrary point, let it be a point on the surface of the planetary body. Given that α can be any value greater than 0, any point (except $(0,0,0)$) along the radial line that extends from the origin through *pos* will yield the same horizon vector. Finally, because *pos* is arbitrary, it may be any point on the surface of the planetary body. Thus, any point in space, with the exception of $(0,0,0)$, will return the appropriate horizon vector. This means that for any node at position (a,b,c), an appropriate position horizon position to look toward will take the simple form (a,b,c) + (1, –*a*/*b*, 0).

One edge case must be considered. When $b=0$, or the point lies on the x-axis or the poles, (1, –*a*/*b*, 0) cannot be used because the y component will be undefined. In this case, the vector $(0,1,0)$ should be used. For a point on the x-axis, the equation $x=r$ forms the tangent. This means that any vector of the form $(0,\alpha,0)$ will be tangent, but $(0,1,0)$ is simple.

References

1. The MathWorks, Inc.: What is MATLAB? https://www.mathworks.com/discovery/what-is-matlab.html Accessed Oct. 17, 2023.
2. JSON: Introducing JSON. https://www.json.org/json-en.html Accessed Oct. 17, 2023.
3. Shalkhauser, Lucas D.; Henderson, Eric; and Welch, Bryan W.: On Development of Three-Dimensional Visualization Capabilities in Glenn Research Center Communication Analysis Suite. NASA/TM-20205000040, 2020. https://ntrs.nasa.gov
4. Bloomingdale, Samuel J.; and Welch, Bryan W.: GCAS Visualization Codebase Augmentation Migration to Modern Standards and Feature Enhancements. NASA/TM20230014837, 2024. https://ntrs.nasa.gov
5. McDonald, John: Animated CGEM: Rotation About an Arbitrary Axis. https://diglib.eg.org/bitstream/handle/10.2312/cgems04-11-1367/Rotation_About_an_Arbitrary_Axis.pdf Accessed Oct. 17, 2023.
6. three.js.: Local Rotation Documentation. https://threejs.org/docs/#api/en/core/Object3D.rotateX Accessed Oct. 17, 2023.
7. Strickland, Jonathan. What is a Gimbal—and What Does it Have to do With NASA? How Stuff Works, 2023. https://science.howstuffworks.com/gimbal.htm Accessed Oct. 17, 2023.
8. Shoemake, Ken: Arcball Rotation Control. Academic Press, 1994. https://research.cs.wisc.edu/graphics/Courses/559-f2001/Examples/Gl3D/arcball-gems.pdf Accessed Oct. 17, 2023.
9. Fileformat: WEBM. https://docs.fileformat.com/video/webm/ Accessed Oct. 17, 2023.
10. Fileformat: MP4. https://docs.fileformat.com/video/mp4/ Accessed Oct. 17, 2023.
11. File System Access: API. https://wicg.github.io/file-system-access/ Accessed Oct. 17, 2023.
12. Mozilla: Piping Documentation. https://developer.mozilla.org/en-US/docs/Web/API/ReadableStream/pipeTo Accessed Oct. 17, 2023.
13. Node.js: Piping Documentation. https://nodejs.org/api/stream.html#event-pipe Accessed Oct. 17, 2023.
14. Mozilla: Blob Documentation. https://developer.mozilla.org/en-US/docs/Web/API/Blob Accessed Oct. 17, 2023.
15. Mozilla: HTTP Documentation. https://developer.mozilla.org/en-US/docs/Web/HTTP Accessed Oct. 17, 2023.