**Design Documentation:**

**NASA EVA Path Phase 4**

**Version 3.1**

**Design Architects**

Tristan Arjoon

Mario Curtis

Shane Farmer

Claudel Guembu

Sean Johnson

**SWEN 670 9040 (2188)**

**UMUC Graduate School of Management and Technology**

**Software Engineering Project**

**2 December 2018**

**Fall Semester**

**Table of Contents**

[Revision History 3](#_gjdgxs)

[1. Introduction 4](#_1fob9te)

[1.1 Background 4](#_3znysh7)

[1.2 Intent 4](#_2et92p0)

[2. Utilization Framework 5](#_tyjcwt)

[3. Design 7](#_3dy6vkm)

[4. References 17](#_1t3h5sf)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Author | Date | Description |
| 1.0 | Tristan Arjoon | 10/16/18 | Created & Updated Draft Content |
| 1.1 | Mario Curtis | 10/17/18 | Updated Introduction and Background |
| 1.2 | Tristan Arjoon | 10/19/18 | Updated sections 1,2,3,4 |
| 2.0 | Tristan Arjoon | 11/04/18 | Updated sections 3, Milestone 3 |
| 2.1 | Mario Curtis | 11/11/18 | Linux Alternative |
| 3.0 | Tristan Arjoon | 12/1/18 | Updated Section 4, Milestone 4. Finalized Document |
| 3.1 | Shane Farmer  Mario Curtis | 12/2/18 | Added steps for Git LFS; minor edits |

**NOTE: Future phases, please jump to Section 3, “Phase 4 Milestone 4”, for an update on where phase 4 left off. Section 2 is a great summary of how to get everything up and running (see manual for more detailed steps). Once everything is installed, “run\_nasa.bat” in the ‘nasa-path-finder’ folder will compile and load everything.**

# 1. Introduction

The purpose of this design documentation is to develop a comprehensive strategy that is required in developing, modifying, or upgrading the NASA Path software program. The document provides aspects of each step and its required deliverables. The design covers every aspect from the project’s process of software development, documentation, through the software testing lifecycle and deployment.

## 1.1 Background

Users of the software are astronauts and the people who train them and write their spacewalk procedures. The software will be used to train astronauts on how to perform spacewalks, plan astronaut spacewalks, and write astronauts procedures.

Also, the software will be used to map out translation paths and choreograph maneuvers involving robotic arm (SSRMS). The software will also be used to avoid hazards, such as radiating antennas and sharp edges. Finally, the software will be used real-time when deviations to the timeline must be made, whether due to contingencies or timeline constraints. The software data input takes input from the user to establish the ideal path(s) for the astronaut. The software data output displays output data of the determined ideal path(s) for the astronaut, then outputs the data so that “DOUG” can import it. Finally, the software data output displays with the application.

## 1.2 Intent

The intent of this document is to give stakeholders a walkthrough of understanding. Previous and current teams are developing code in “chunks” or cycles. This document will allow users to understand how to analyze the software DOUG, all relevant files, and the test environment user interface. By exposing a strategy that demonstrates the process by which a solution is being achieved, one that marries different variables from designthrough development and implementation.

Phase 4’s main objective is to illustrate the design:

* Add indications of when axial direction and plane changes occur
  + e.g. from port to zenith or from face 1 to face 2
* Provide a legend/key on how to maneuver ISS
* Expand the code to make use of the entire ISS model
  + This is the highest priority item per the client

These objectives will be outlined during section 3 after the document explains the initial framework that needs to be established in section 2.

# 2. Utilization Framework

Before understanding each file, a framework by all developers and testers must be established. Below is a detailed list of various software needed to establish a working EVA Path working environment. For more detail on each software specifications, refer to User Manual established by Phase 3.

Step 1:

Install Node.js

* JavaScript runtime for Chrome’s JS engine.

Step 2:

Install Yarn

* Package manager for all localized code.

Step 3:

Install Java (minimum version 8)

* Object oriented language & environment

Step 4:

Install Visual Studio

* An Integrated Development Environment (IDE).

Step 5:

Install Python (Not needed for to see working environment)

* Integrated programming language.

Install Maven (Not needed for to see working environment)

* Project Management tool for software documentation & reporting.

Step 6:

Install Git LFS (needed for large model files)

Step 7:

Visit GitHub repository established by the customer for latest code and builds.

Step 8:

In the repository is a file called “run\_nasa” the batch file for windows users. This file will initiate above software to localize a web server, package code, and establish it in Chrome under [http://127.0.0.1:3000](http://127.0.0.1:3000/) or<http://localhost:3000>. See Section 5 of the manual for more information.

Step 9:

Get access to DOUG (Dynamic Onboard Ubiquitous Graphics Software)

See User Manual under documents for more information on how to get access.

**LINUX Alternative:**

*DockerFile Image* - Allows Container to pull dependencies for the program

Step 1:

In the Command Terminal Type:

sudo apt-get install update && sudo apt-get install upgrade -y

sudo apt-get install docker

sudo apt-get install docker-compose

Step 2:

In the Command Terminal Type:

cd

cd /Desktop

mkdir Source

cd Source

Step 3:

curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash

sudo apt-get install git-lfs

Step 4:

In the Command Terminal Type

sudo docker run -p 8080:8080 -p 3000:3000 lovetostrike/nasa-path-builder

# 3. Design

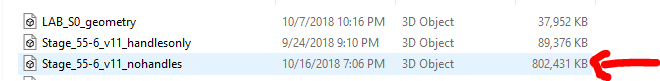
**Phase 4 Milestone 1:**

During this phase all software was established as illustrated above.

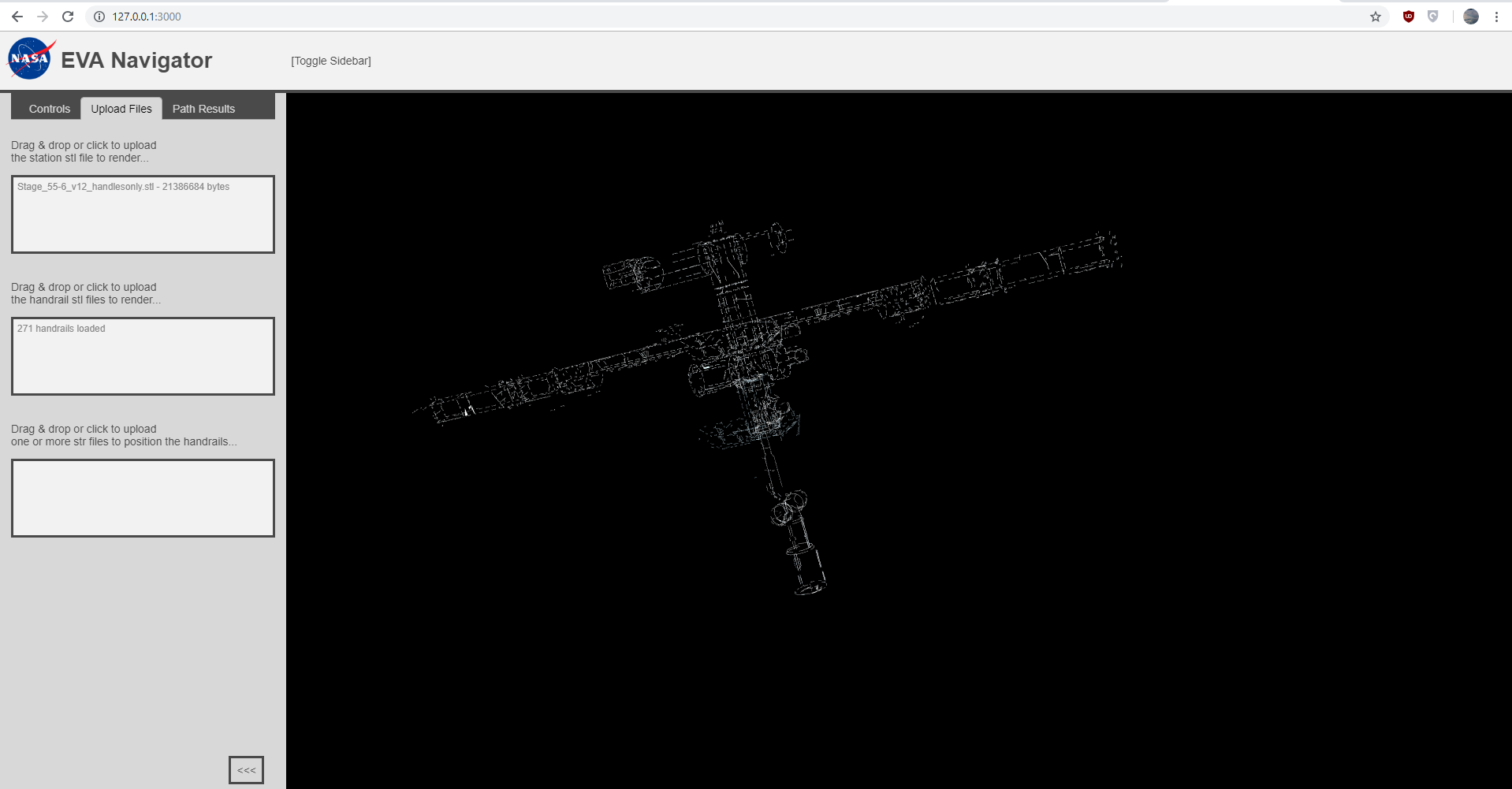
**Phase 4 Milestone 2:**

Phase 4 decided to tackle the customer’s highest priority issue. Establishing the entire ISS model in the browser.

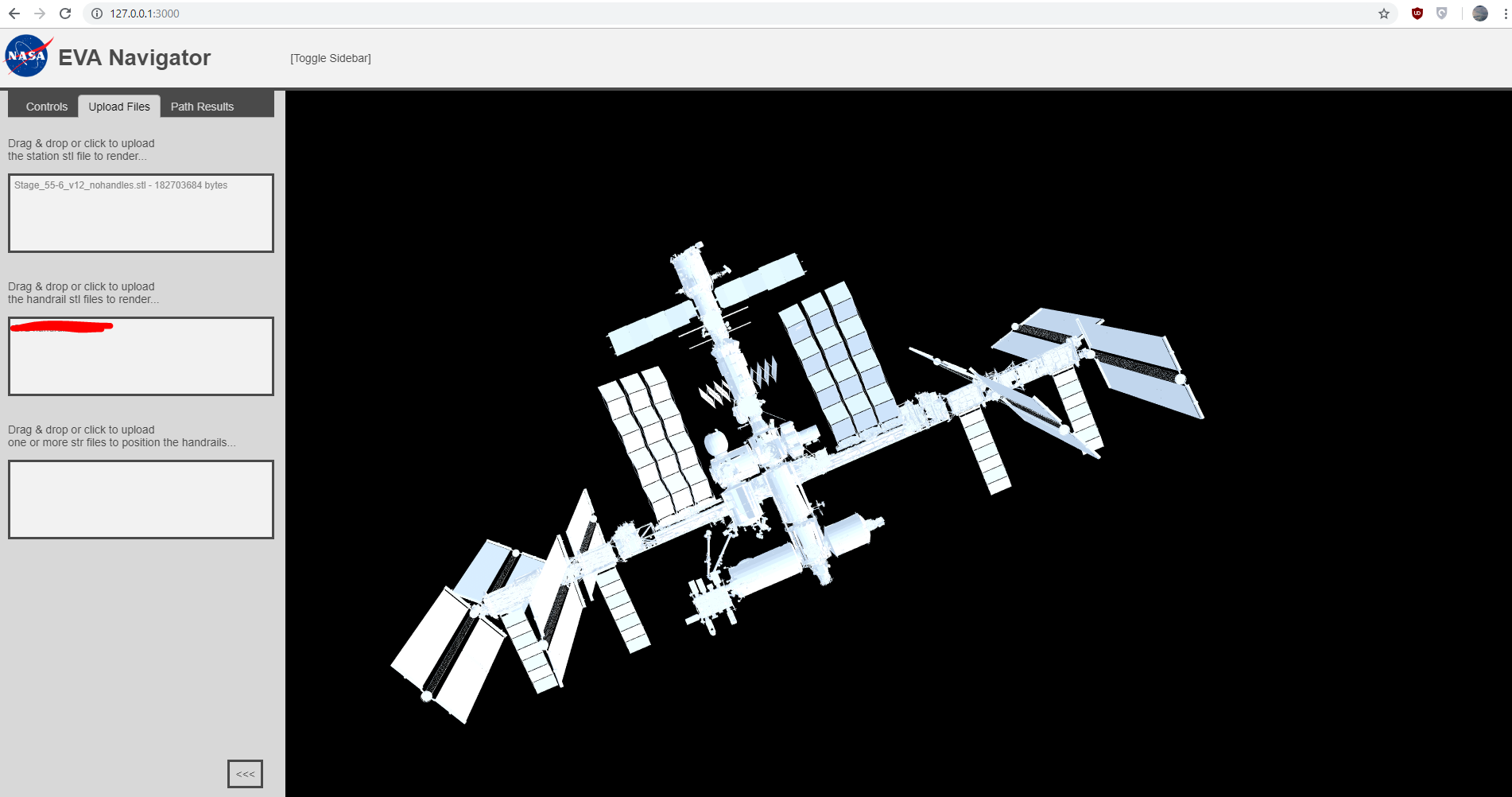
Previous teams had trouble with loading the current stage (Stage\_55-6\_v11\_nohandles.stl) into the browser. The file was 800MBs and caused the browser to crash.



An updated .stl file (Stage\_55-6\_v12\_nohandles.stl) was established in the \nasa-path-finder-master\SampleData directory (see github). This file eliminates the timeout (crash) issue previous phases had by converting the internals to binary using Blender (see section 2 for more information on Blender). The file is now 170 MBs and loads properly into the browser. See below for 2 pictures showing just handles & ISS model



User interface showing handles of the whole ISS model

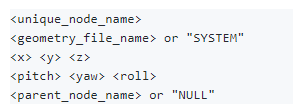


User interface showing whole ISS model

**Phase 4 Milestone 3:**

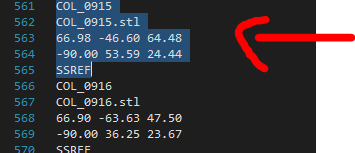
**Problem:** For this phase, the team worked on several issues that arose from the previous milestone. One of the biggest issues was the alignment of the handrails to the newly crafted STL document that incorporates the whole ISS without stalling issues. Furthermore, the team needs updated .str information from DOUG dev team.

**Solution:** To be able to utilize the handrails that are pulled from DOUG, the team has sent out an email to the customer requesting all handrail frames to be separate .stl files. Now, the team is currently waiting on a separate email from the customer and DOUGs dev team to hopefully get DOUG data exported to a file. This data the team is currently waiting for is illustrated below:



Pseudocode of each segment of the .str file needed

Then we can parse that data into the “Entire\_ISS.str file for accurate locations, pitch, yaw, and roll. This information is currently and manually being entered by the team for testing. Once we can extract that information, can the ISS have all handrails in correct formation to be aligned to the ISS.



Segment showing proper handrail configuration (currently manually input)

Below is an image of DOUG showing the information we need to be exported as a file to be parsed into .str file.



Image of DOUG showing data needed.

Currently, the ISS handrails STL files are being loaded as “xyz = 0” due to .str segments not appropriately updated until DOUG devs respond with extraction data.

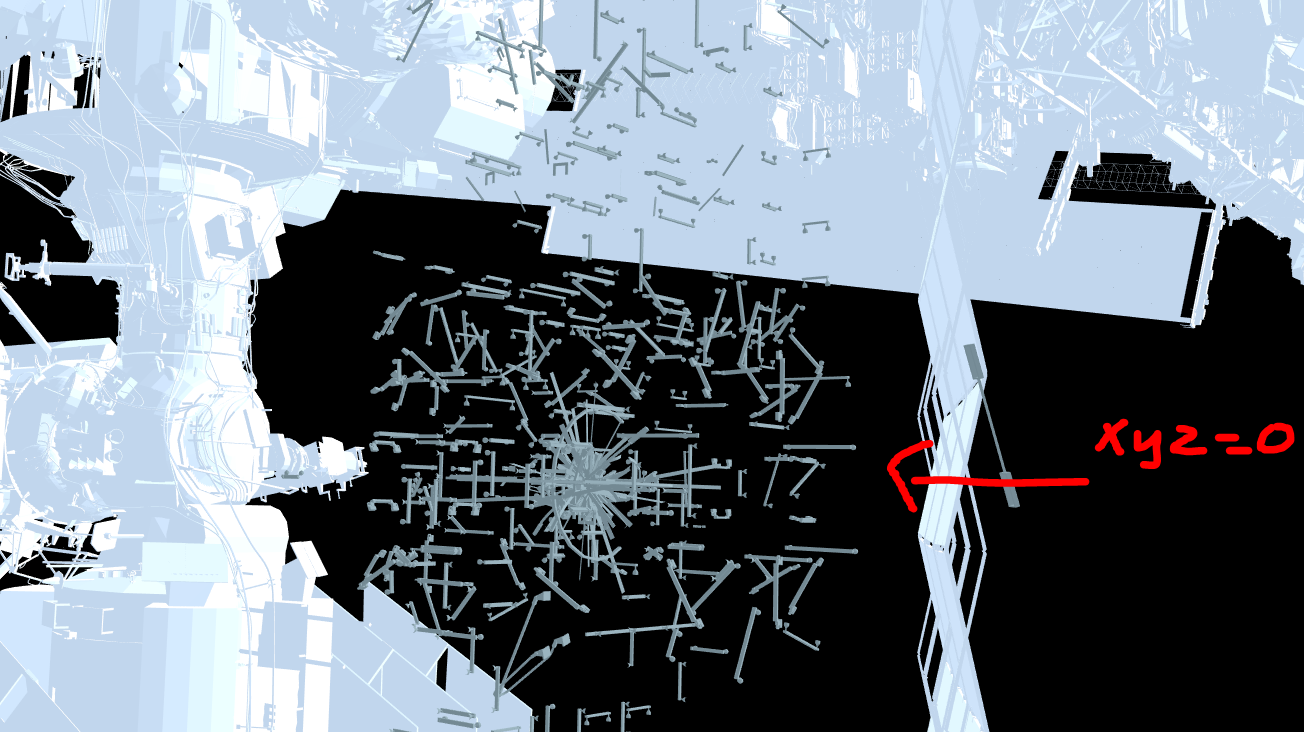
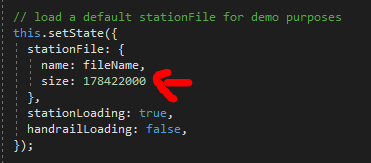


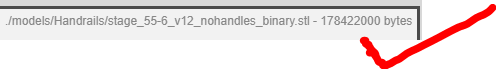
Illustration showing axis’s needed from DOUG most represented as 0 currently

**Other Fixes:** Advancements were made to the Controls file. This file is found in, “\src\components\Controls Controls.js”. Redesign of this code allowed for:

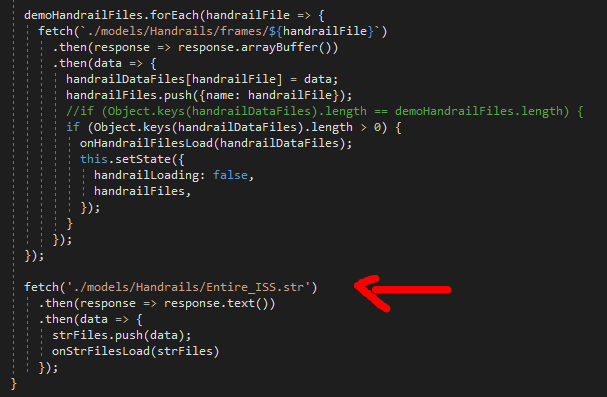
* + Accurate Byte calculation
  + File fetching
  + Naming convention
  + Redundant code removal

See below for illustrated changes:





Reflects ISS STL file size (should not change, hence static code)



Reflects updated file paths for fetching files automatically upon UI boot.

**Phase 4 Milestone 4:**

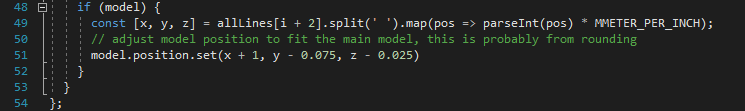
**Problem:** This milestone will solve several NEWLY found issues. and will be updated for next deliverable.

* Incorporate handrails appropriately (location, haw, pitch, roll)
* Align handles to ISS model using STR files for coordinates
* Incorporate an ISS UI legend
* ISS was too bright

**Solution:** Phase 4 successfully solved all problems stated above. As well as implemented some features. See below for additions.

Aligned Handrails to ISS Appropriately

One of our phases biggest accomplishments was getting the entire ISS to load in the UI! However, this created problems with location, yaw, pitch, and roll stats that were previously downloaded from DOUG. After working with the customer (NASA) and their vendors (CACI) to retrieve proper locations of handrails that matches “.stl” names, we were able to manipulate and extract that data and align the handrails to the entire ISS utilizing the below code modifications.



Code addition to fix rounding issue (src utils NodeProcessor.js)

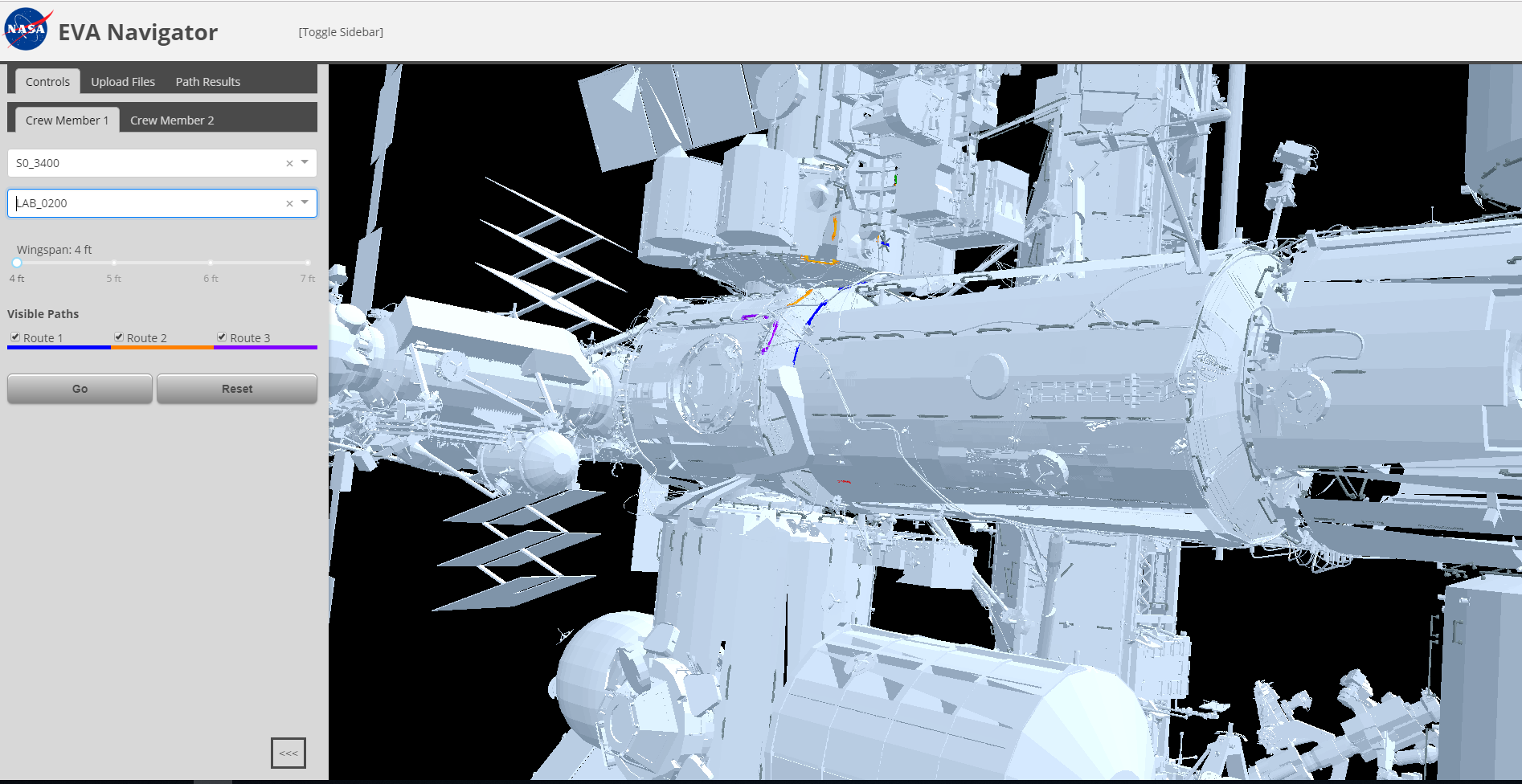


Image of entire ISS with handrails flushed appropriately

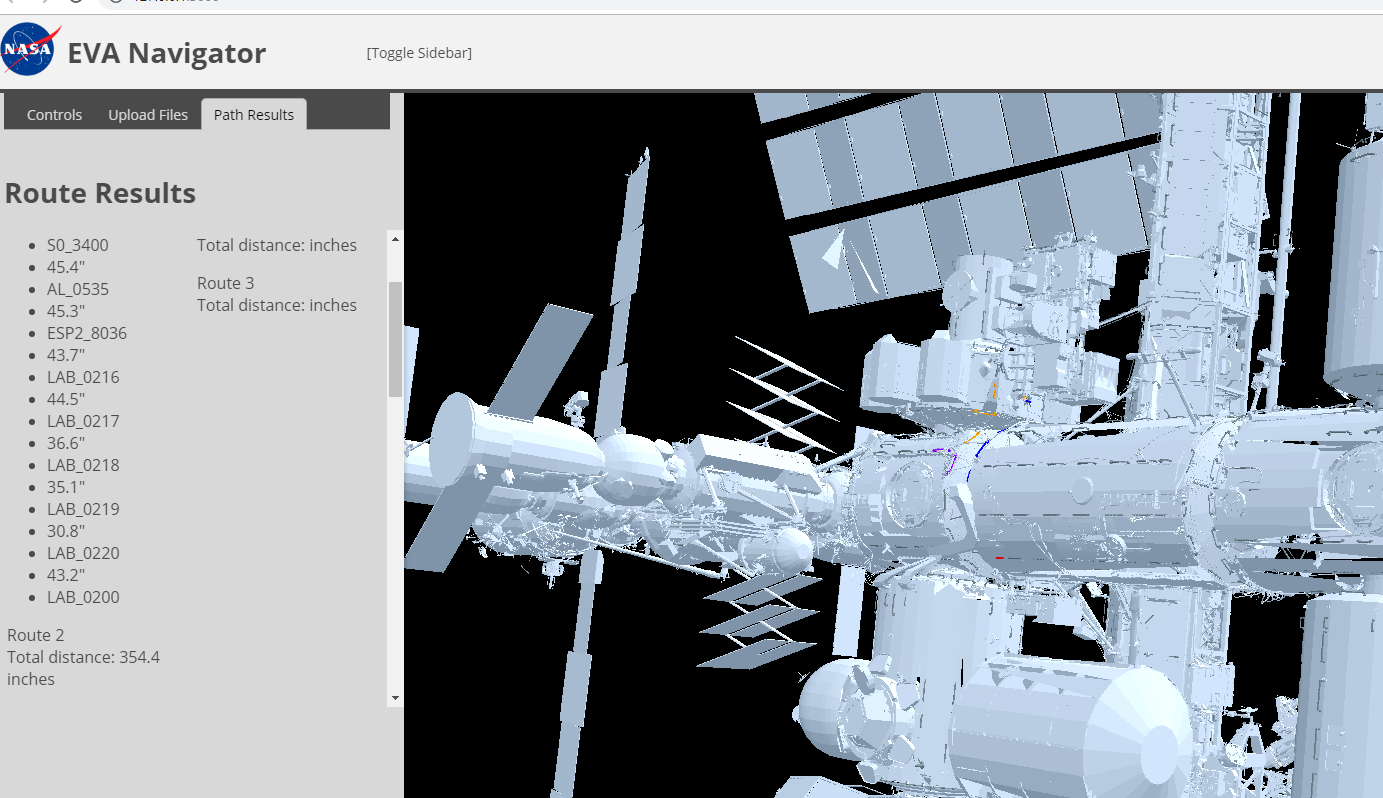
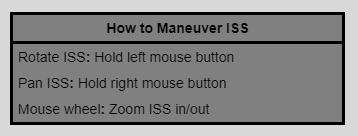


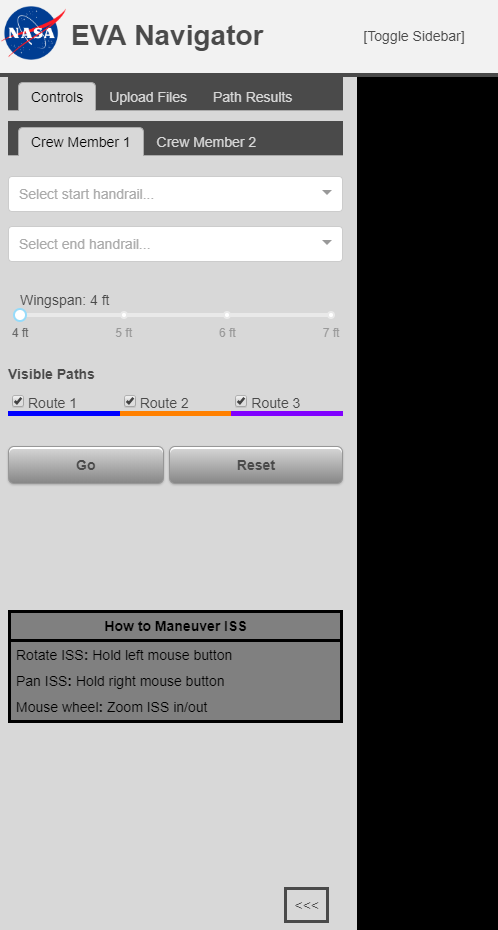
Image showing path/handrail functionality

Implemented a UI Legend

A legend was implemented in the UI to show how the user can maneuver the entire ISS. Code can be found and tweaked in (src components Controls.js) and (src \_layout.scss). The legend can be found under both crew member tabs:



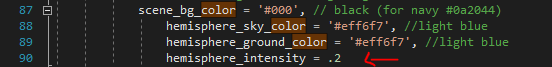
ISS UI Legend



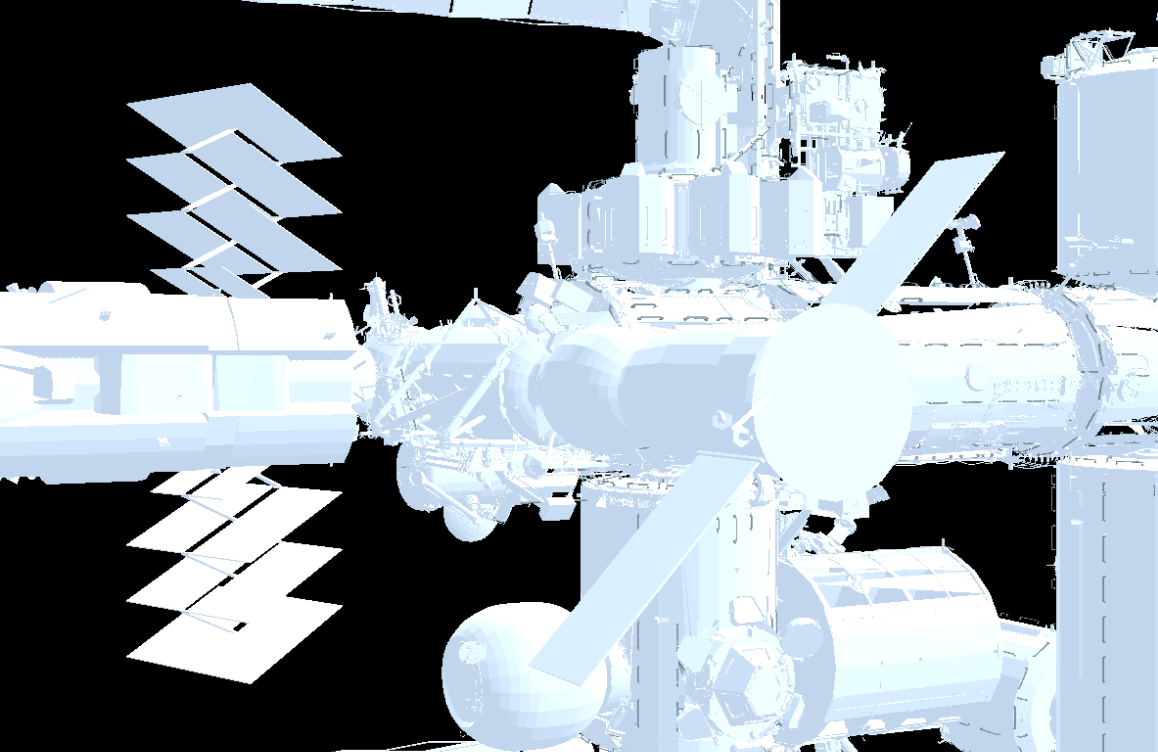
Placement of legend in UI

Altered Brightness of the ISS

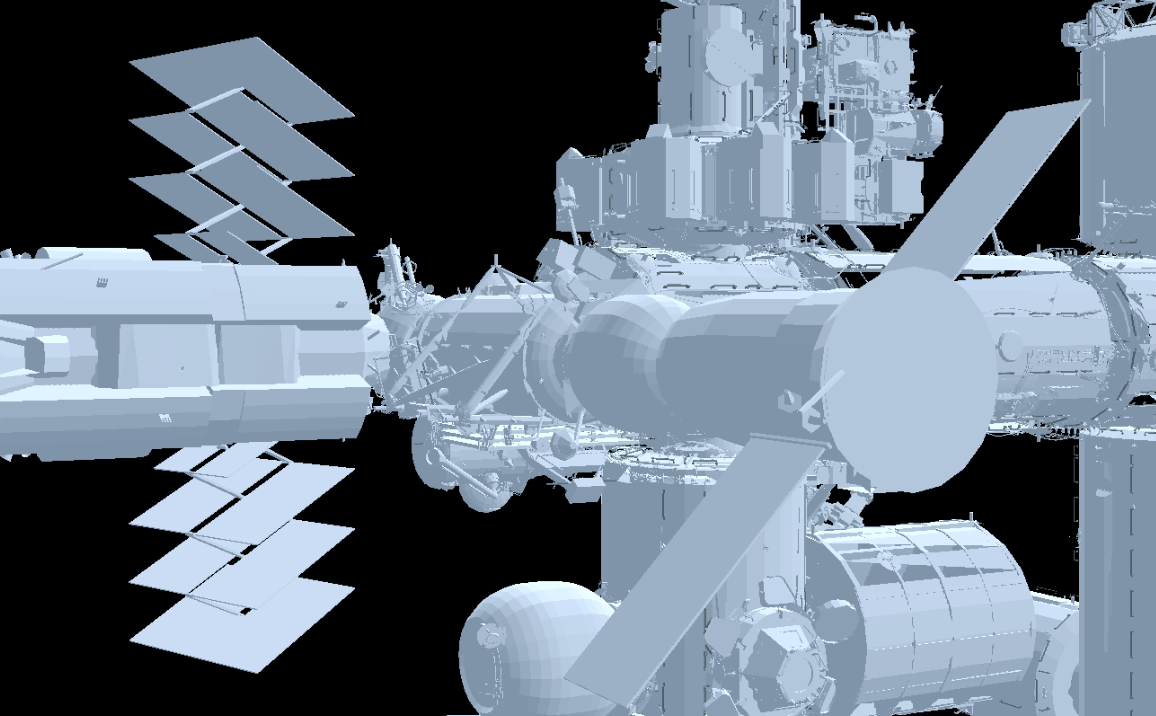
The previous ISS was way too bright against the darkness of space (UI background) so phase 4 team decided to adjust the brightness appropriately. Once entire ISS loaded without issues. See below for code, and before & after images.



Snapshot of code in (src Render.js)



BEFORE



AFTER

# 4. References

* Guembu, C. (2018). Navigation for space walks. Retrieved from <https://github.com/claudelguembu/nasa-path-finder>