Feature Documentation: Handrail Selection using ISS Model

for

NASA EVA Path Phase 3

Version 1.1

Developer
Lincoln Powell

Product Owner
Daren Welsh

Test Team
Tenadam Weldesemayat

University of Maryland University College
SWEN 670 9040 (2185)
Software Engineering Project
Dr. Michael Brown

August 11, 2018

Table of Contents

i. Revision History

Revision	Author	Date	Description
1.0	Lincoln Powell	8/8/18	Initial document.
1.1	Tenadam Weldesemayat	8/11/18	Reviewed document and added as part of the test team.

1. Introduction

1.1 Background

The finalized NASA EVA Navigator web application delivered from Phase 2 completed tailored backlog items designed by Daren to meet existing objectives for the product, one being the selection of start and end points, or handrails, on the ISS model using dropdown lists. Although the dropdown lists were sufficient to meet the needs for the requirement, Daren sought to emulate the Dynamic Onboard Ubiquitous Graphics (DOUG) software by extending the start and end point selection of the application to allow users to click desired handrails on the International Space Station (ISS) model.

1.2 Intent

Daren's intent for this change is to allow users to click a handrail in the ISS model to select start and end points for route calculation.

1.3 Agreed Change(s)

Lincoln accepted Daren's requested change as-is. Application feel in regards to users selecting start or end points using the dropdown lists should be preserved (e.g. once a start or end handrail is chosen, it must be deselected before choosing a new handrail).

2. Development

2.1 Code Additions or Modifications

In order to interact with the ISS 3D model using WebGL, ray casting needed to be performed from the mouse cursor to the handrail mesh to select the corresponding handrail object in the respective dropdown list. More, since the dropdown lists are React-Select select components, event and state management was heeded.

In the Renderer.js file of the EVA Navigator project, a global Raycaster object, *raycaster*, is declared and initialized. Class attributes, *mouse* and *camera*, were modified as global objects for definition outside the class context.

```
26 // PHASE 3 MOD Lincoln Powell/lpowell125@student.umuc.edu 8/3/2018
27 // Declare and initialize Raycaster object.
28 var raycaster = new THREE.Raycaster();
29
30 //PHASE 3 MOD Lincoln Powell/lpowell125@student.umuc.edu 8/3/2018
31 // Declare and initialize Vector2 object representing the 2D vector of a mouse cursor.
32 var mouse = new THREE.Vector2();
33
34 //PHASE 3 MOD Lincoln Powell/lpowell125@student.umuc.edu 8/3/2018
35 // Declare and initialize PerspectiveCamera object.
36 var camera = new THREE.PerspectiveCamera(35, window.innerWidth / window.innerHeight, 0.0001, 5000);
```

Figure 1. *Raycaster*, *mouse* and *camera* object declarations and initializations.

An empty global array, *handrailMeshes*, is also declared and initialized to be used to get all handrail meshes to be used to allow for ray casting.

```
46 //PHASE 3 MOD Lincoln Powell/lpowell25@student.umuc.edu 8/3/2018
47 // Declare empty array for handrail meshes used to raycast the mouse cursor to each handrail on the model.
48 var handrailMeshes = [];
```

Figure 2. Creating an empty array, *handrailMeshes*, to hold all handrail meshes for ray casting.

A function was created, *onDocumentMouseDown*, to fire when a user clicks down on a handrail, which orients the Raycaster object from the camera object's position to the mouse cursor then evaluates if an intersection occurred between the mouse cursor and a handrail mesh. If an intersection occurred and, as long as a start or end handrail has not been chosen (meaning the start or end handrail dropdown selection equates to null), call the Container.js *handleStartEndHandrailsChanged(String, Object, Boolean)* handler to fire the onChange event; thus, populating the respective dropdown list. If the chosen handrail is the preselected start or end handrail, the respective handrail is cleared.

```
164 /**
165  * PHASE 3 MOD
166  * @author Lincoln Powell/lpowell25@student.umuc.edu
167  * @since 8/3/2018
168  *
169  * The onDocumentMouseDown function orients the Raycaster, raycaster, object from the camera to the mouse cursor
170  * then evaluates if an intersection occurred between the mouse cursor and a handrail on the event a user
171  * clicking on a handrail.  If an intersection occurred and as long as a start or end handrail has not been chosen
172  * (meaning the start or end handrail dropdown selection equates to null), call the handleStartEndHandrailsChanged(string, object, boolean)
173  * handler in the Container.js to fire the onChange event; thus, populating the respective dropdown.
174  */
175 onDocumentMouseDown = (e) => {
176     e.preventDefault();
177     mouse.x = (e.clientX / window.innerWidth) * 2 - 1;
178     mouse.y = -(e.clientY / window.innerHeight) * 2 + 1;
179     raycaster.setFromCamera(mouse, camera);
180     var intersects = raycaster.intersectObjects(handrailMeshes);
181     if (intersects.length > 0) {
182 
183         // If the start handrail or the end handrail equals null (meaning neither handrail has been chosen)
184         // or if the chosen handrail is not the preselected start or end handrail, mark the handrail respectively.
185         if ((!this.props.startHandrail || !this.props.endHandrail)
186                 && this.props.startHandrail != intersects[0].object.name.replace('.stl', '')
187                 && this.props.endHandrail != intersects[0].object.name.replace('.stl', '')) {
188             this.props.handleStartEndHandrailsChanged(this.props.startHandrail?'end':'start', intersects[0].object, true);
189         }
190 
191         // Else if the chosen handrail is the preselected start handrail, clear the start handrail.
192         else if (this.props.startHandrail == intersects[0].object.name.replace('.stl', '')) {
193             this.props.handleStartEndHandrailsChanged('start', null, false);
194         }
195 
196         // Else if the chosen handrail is the preselected end handrail, clear the end handrail.
197         else if (this.props.endHandrail == intersects[0].object.name.replace('.stl', '')) {
198             this.props.handleStartEndHandrailsChanged('end', null, false);
199         }
200     }
201 }
```

Figure 3. *onDocumentMouseDown* function.

A function was created, *onDocumentMouseMove*, to fire when a user moves the mouse cursor over a handrail, which orients the Raycaster object from the camera object's position to the mouse cursor then evaluates if an intersection occurred between the mouse cursor and a handrail mesh. If an intersection occurred, the mouse cursor will be changed to a pointer; else, the mouse cursor will be the default cursor.

```
203  /**
204   * PHASE 3 MOD
205   * @author Lincoln Powell/lpowell25@student.umuc.edu
206   * @since 8/3/2018
207   *
208   * The onDocumentMouseMove function orients the Raycaster, raycaster, object from the camera to the mouse cursor
209   * then evaluates if an intersection occurred between the mouse cursor and a handrail on the event a user
210   * moves the cursor over a handrail.  If an intersection occurred, the mouse cursor will be changed to a pointer; else,
211   * the mouse cursor will be the default cursor.
212   */
213  onDocumentMouseMove(e) {
214      e.preventDefault();
215      mouse.x = (e.clientX / window.innerWidth) * 2 - 1;
216      mouse.y = -(e.clientY / window.innerHeight) * 2 + 1;
217      raycaster.setFromCamera(mouse, camera);
218      var intersects = raycaster.intersectObjects(handrailMeshes);
219      var canvas = document.body.getElementsByTagName('canvas')[0];
220      if (intersects.length > 0) {
221          canvas.style.cursor = 'pointer';
222      } else {
223          canvas.style.cursor = 'default';
224      }
225  }
```

Figure 4. *onDocumentMouseMove* function.

In the *processFiles* function, for each handrail read from file, its mesh is added to the array.

```
317      if (handrailFiles && Object.keys(handrailFiles).length > 0 && strFiles && strFiles.length > 0 ) {
318          Object.values(this.handrailModels).forEach(model => this.scene.remove(model));
319          Object.entries(handrailFiles).forEach(([name, handrailFile]) => {
320              let color = hrColor;
321              let scale = 1;
322              var handrailMesh = loadMeshFromFile(handrailFile, {color}, {scale});
323
324              // PHASE 3 MOD Lincoln Powell/lpowell25@student.umuc.edu 8/3/2018
325              // Add each handrail mesh to the handrailMeshes array.  This array is used to raycast the mouse cursor to each handrail on the model.
326              handrailMeshes.push(handrailMesh);
```

Figure 5. Adding each handrail mesh to the *handrailMeshes* array (line 326).

In the Container.js file of the EVA Navigator project, the *handleStartEndHandrailsChanged* handler was expanded to set state of start or end dropdown values beyond the use of the dropdown lists. A new Boolean parameter was added to the handler, *clickedHandrail*, to determine if the onChange event occurred from a user clicking on a handrail. By default, the parameter is false; however, on the event a user clicks on a handrail, a new handrail object is created, setting its value to the name of the clicked handrail. Then, the state is set of the start or end handrail dropdown. Finally, the Renderer.js *componentDidUpdate* function is fired, coloring the start or end handrail appropriately. This function is then passed as a property to the Renderer class to be freely called.

```
/**
 * PHASE 3 MOD
 * @author Lincoln Powell/lpowell25@student.umuc.edu
 * @since 8/9/2018
 *
 * The handleStartEndHandrailsChanged(string, object, boolean) handler controls when the onChange event is fired
 * to set the start or end handrail dropdown value.  If user clicked on a handrail, create a new handrail object,
 * setting its value to the name of the handrail.  Then set the state of the start or end handrail dropdown.
 * Finally, the Renderer.js componentDidUpdate() function is called, coloring the start or end handrail appropriately.
 */
handleStartEndHandrailsChanged(startOrEnd, handrail, clickedHandrail = false) {
    if (clickedHandrail) {
        handrail = { value: handrail.name.replace(".stl","") }
    }

    this.setState({
        [`${startOrEnd}Handrail`]: handrail?handrail.value:null
    });
}
```

Figure 6. *handleStartEndHandrailsChanged* handler.

The *startHandrail* and *endHandrail* properties were changed from Object to String, as it did not make sense to have a dropdown list of objects; as such, the *handleSubmit* function was modified where the properties are used. If the *startHandrail* or *endHandrail* are not null, pass its value instead of the previously used Object.value; else, pass in an empty String.

3. Functional Testing

3.1 Proposed Functional Test Cases

3.1.1 Test Case 1: Select starting handrail on ISS model

Description: The user should see a green handrail and the start handrail drop-down menu populated with handrail name when a starting handrail is selected from the ISS model.
Requirements:

1. User should be able to choose a starting handrail by clicking on a handrail on the ISS model.

Prerequisites: None.
Steps:

1. Load “EVA Navigator” web application by navigating browser to http://127.0.0.1:3000 or http://localhost:3000.
2. Select the LAB_0200 handrail on the ISS model as the starting handrail.
3. Check that the LAB_0200 handrail on the ISS model is green.
4. Check that the start handrail drop-down menu displays “LAB_0200”.

Input: Click on the LAB_0200 handrail on the ISS model.
Expected Output: Handrail, LAB_0200, on the ISS model is green and start handrail drop-down menu displays “LAB_0200”.
Assumptions: None.

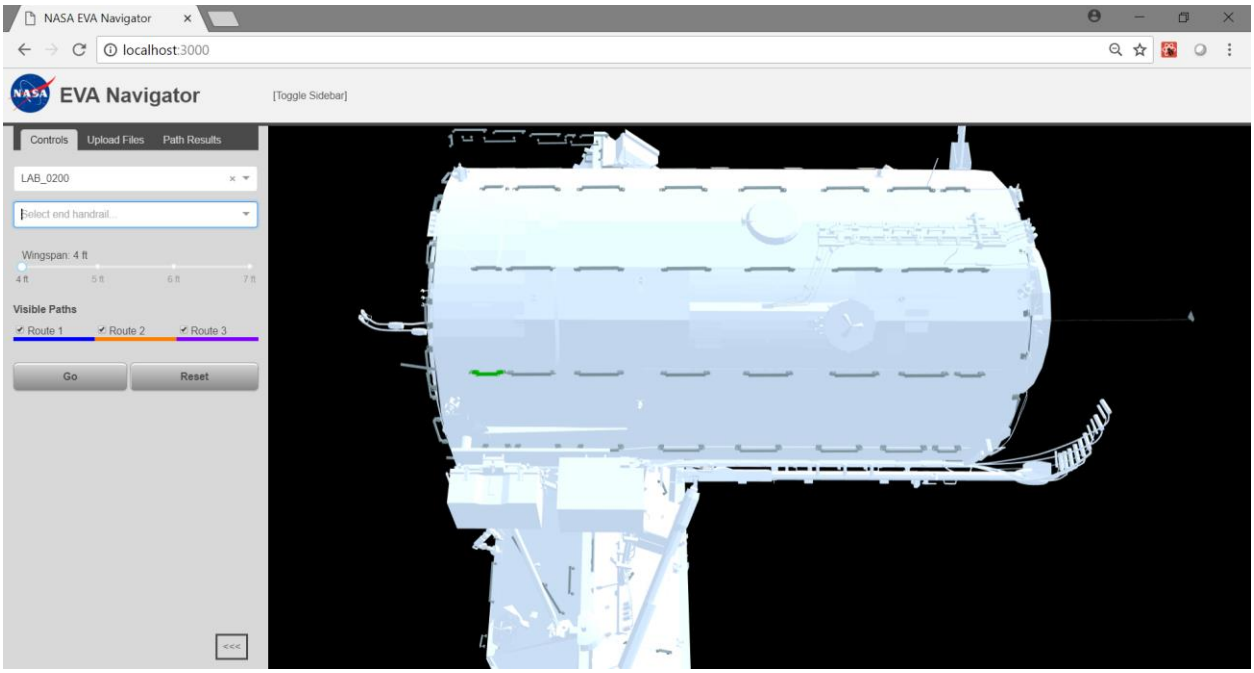

Figure 7. Test case 1 displaying clicked handrail, LAB_0200, as green and its name updated in the start handrail dropdown list.

3.1.2 Test Case 2: Select ending handrail on ISS model

Description: The user should see a red handrail and the end handrail drop-down menu populated with handrail name when an ending handrail is selected from the ISS model.
Requirements:

1. User should be able to choose an ending handrail by clicking on a handrail on the ISS model.
2. User should only be able to choose an ending handrail if a starting handrail was chosen.

Prerequisites:

3.1.1 Test Case 1: Select starting handrail on ISS model

Steps:

1. Load “EVA Navigator” web application by navigating browser to http://127.0.0.1:3000 or http://localhost:3000.
2. Select the LAB_0200 handrail on the ISS model as the starting handrail.
3. Select the LAB_0207 handrail on the ISS model as the ending handrail.
4. Check that the LAB_0207 handrail on the ISS model is red.
5. Check that the end handrail drop-down menu displays “LAB_0207”.

Input: Click on the LAB_0200 handrail then the LAB_0207 handrail on the ISS model.
Expected Output: Handrail, LAB_0207, on the ISS model is red and end handrail drop-down menu displays “LAB_0207”.
Assumptions: None.

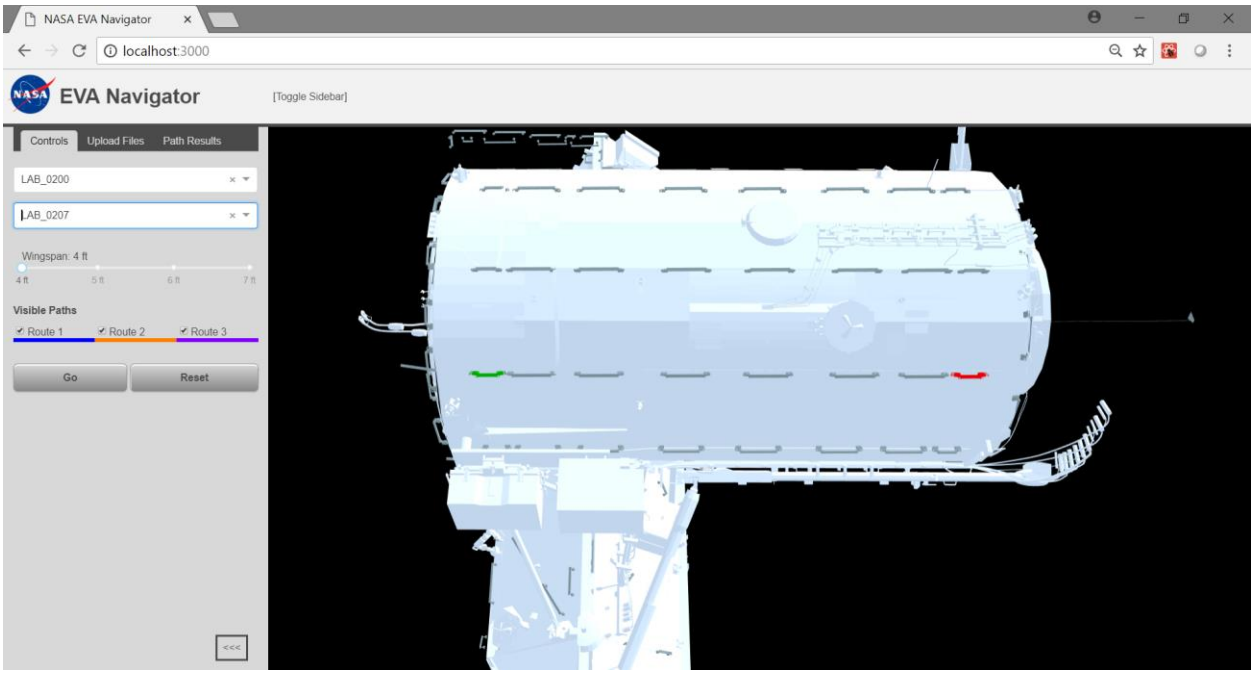

Figure 8. Test case 2 displaying clicked handrails, LAB_0200 and LAB_0207, as green for start and red for end; additionally, each corresponding dropdown list values are updated.

3.1.3 Test Case 3: Clearing starting handrail on ISS model by clicking on the handrail

Description: The user should be able to clear the preselected starting handrail by clicking on the handrail on the ISS model.

Requirements:

1. User should be able to clear preselected starting handrail by clicking on the handrail on the ISS model.

Prerequisites:

3.1.1 Test Case 1: Select starting handrail on ISS model

Steps:

1. Load "EVA Navigator" web application by navigating browser to http://127.0.0.1:3000 or http://localhost:3000.
2. Select the LAB_0200 handrail on the ISS model as the starting handrail.
3. Select the LAB_0200 handrail on the ISS model again to clear the selection.
4. Check that the LAB_0200 handrail on the ISS model is grey.
5. Check that the start handrail drop-down menu displays "Select start handrail…".

Input: Click on the LAB_0200 handrail on the ISS model then click on the LAB_0200 handrail again.

Expected Output: Handrail, LAB_0200, on the ISS model is grey, indicating it is deselected. Start handrail drop-down menu displays "Select start handrail…" placeholder.

Assumptions: None.

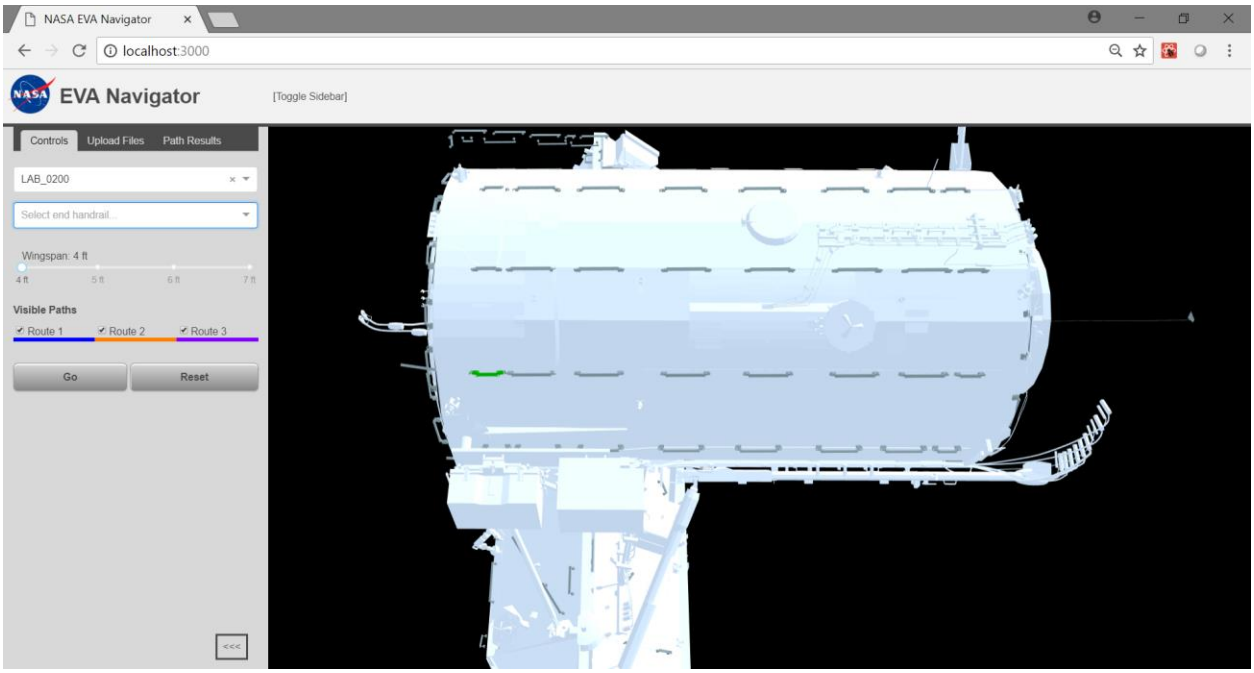

Figure 9. Test case 3 displaying clicked handrail, LAB_0200.

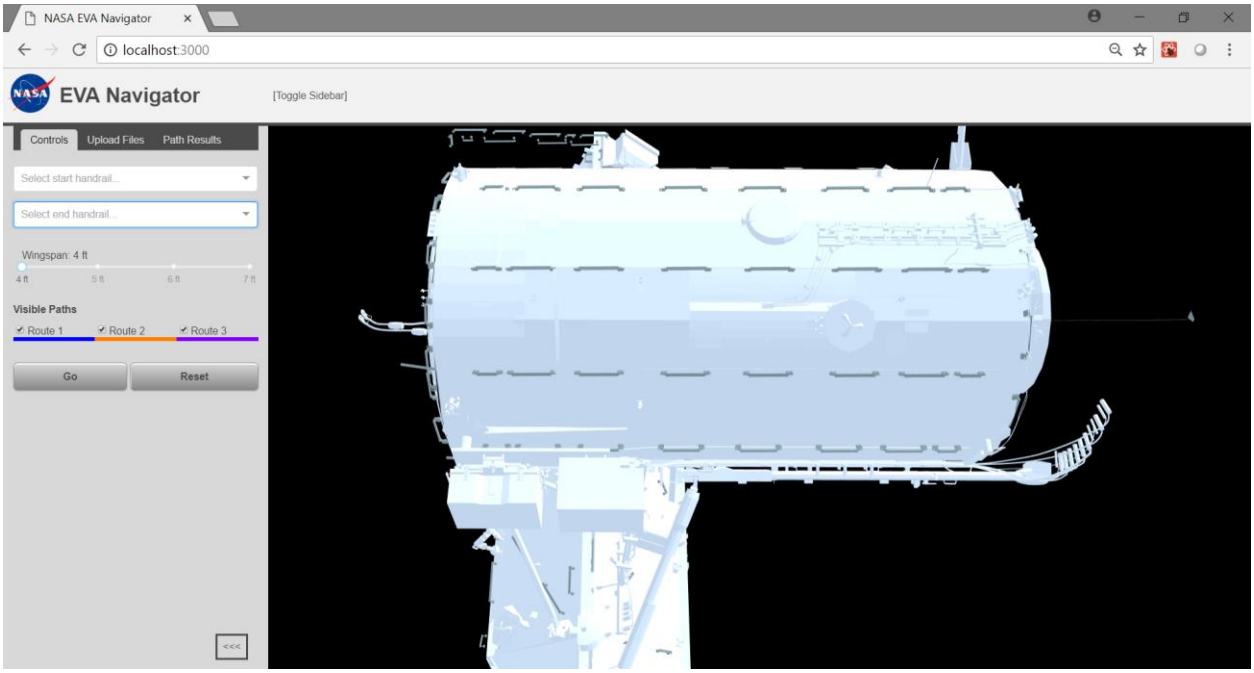

Figure 10. Test case 3 displaying deselected handrail, LAB_0200, after clicking on the handrail again; additionally, notice the start handrail dropdown is now empty with placeholder “Select start handrail…”.

3.1.4 Test Case 4: Clearing ending handrail on ISS model by clicking on the handrail

Description: The user should be able to clear the preselected ending handrail by clicking on the handrail on the ISS model.

Requirements:

1. User should be able to clear preselected ending handrail by clicking on the handrail on the ISS model.

Prerequisites:

3.1.1 Test Case 1: Select starting handrail on ISS model

3.1.2 Test Case 2: Select ending handrail on ISS model

Steps:

1. Load “EVA Navigator” web application by navigating browser to http://127.0.0.1:3000 or http://localhost:3000.
2. Select the LAB_0200 handrail on the ISS model as the starting handrail.
3. Select the LAB_0207 handrail on the ISS model as the ending handrail.
4. Select the LAB_0207 handrail on the ISS model again to clear the selection.
5. Check that the LAB_0207 handrail on the ISS model is grey.
6. Check that the end handrail drop-down menu displays “Select end handrail...”.

Input: Click on the LAB_0200 handrail on the ISS model. Click on the LAB_0207 handrail then click on the LAB_0207 again.

Expected Output: Handrail, LAB_0207, on the ISS model is grey, indicating it is deselected. End handrail drop-down menu displays “Select end handrail…” placeholder.

Assumptions: None.

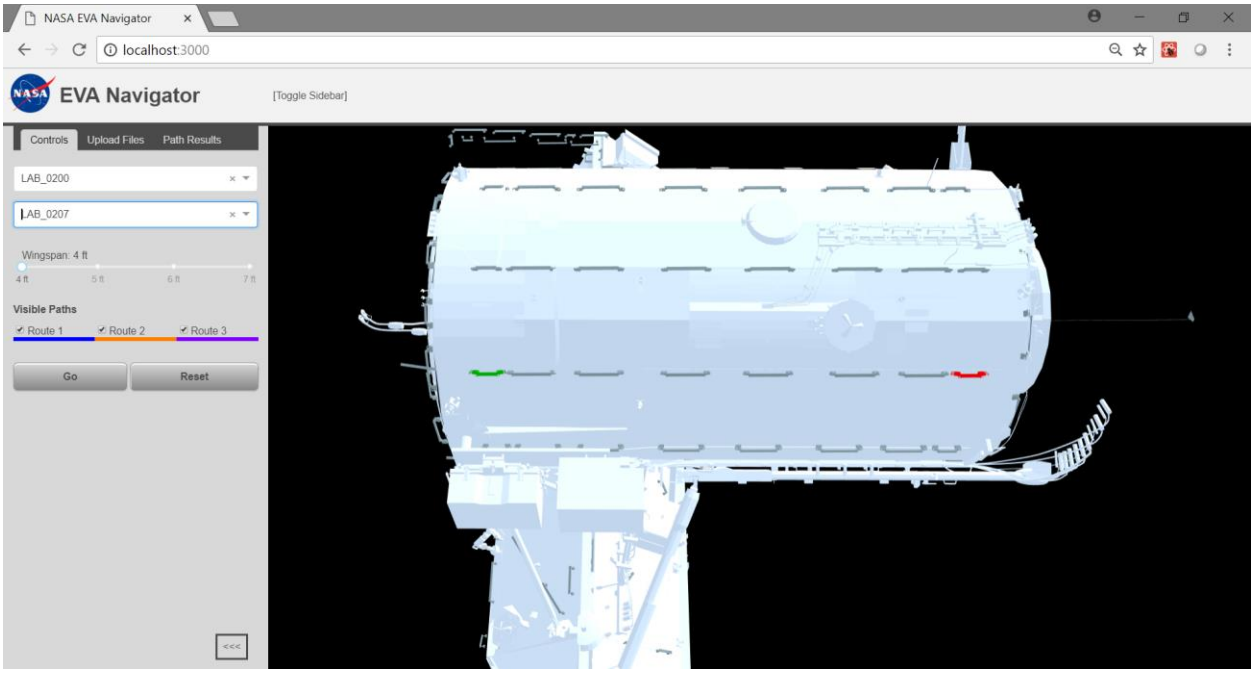

Figure 11. Test case 4 displaying clicked handrails, LAB_0200 and LAB_0207.

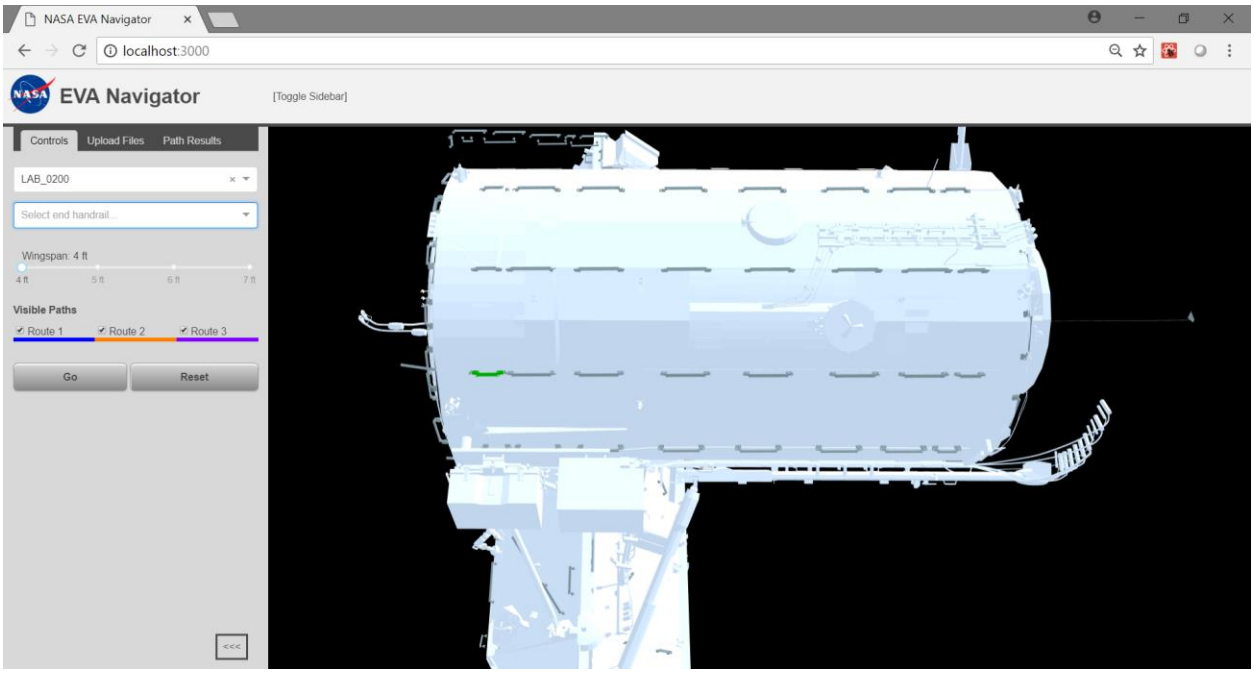

Figure 12. Test case 4 displaying deselected handrail, LAB_0207, after clicking on the handrail again; additionally, notice the end handrail dropdown is now empty with placeholder "Select end handrail…".

3.1.5 Test Case 5: Clearing handrail drop-down selections using ISS model

Description: The user should be able to clear preselected starting and ending handrails chosen via their respective drop-down menus by clicking on the respective handrail on the ISS model.

Requirements:

1. User should be able to clear preselected starting and ending handrails chosen via the drop-down menus by clicking on the handrail on the ISS model.

Prerequisites:

3.1.3 Test Case 3: Clearing starting handrail on ISS model by clicking on the handrail
3.1.4 Test Case 4: Clearing ending handrail on ISS model by clicking on the handrail
Start and end handrail drop-down menus must work as intended.

Steps:

1. Load "EVA Navigator" web application by navigating browser to http://127.0.0.1:3000 or http://localhost:3000.
2. Select LAB_0200 as the starting handrail using the start handrail drop-down menu.
3. Select LAB_0207 as the ending handrail using the end handrail drop-down menu.
4. Click on the LAB_0200 handrail on the ISS model to clear the selection.
5. Check that the LAB_0200 handrail on the ISS model is grey.
6. Check that the start handrail drop-down menu displays "Select start handrail…".
7. Click on the LAB_0207 handrail on the ISS model to clear the selection.
8. Check that the LAB_0207 handrail on the ISS model is grey.
9. Check that the end handrail drop-down menu displays "Select end handrail…".

Input: Click on the starting handrail drop-down menu and select the "LAB_0200" handrail. Click on the ending handrail drop-down menu and select the "LAB_0207" handrail. Click on the LAB_0200 handrail on the ISS model. Click on the LAB_0207 handrail on the ISS model.

Expected Output: Both the LAB_0200 and LAB_0207 handrails are grey, indicating they are deselected. The start handrail drop-down menu displays “Select start handrail…”. The end handrail drop-down menu displays “Select end handrail…”.
Assumptions: None.

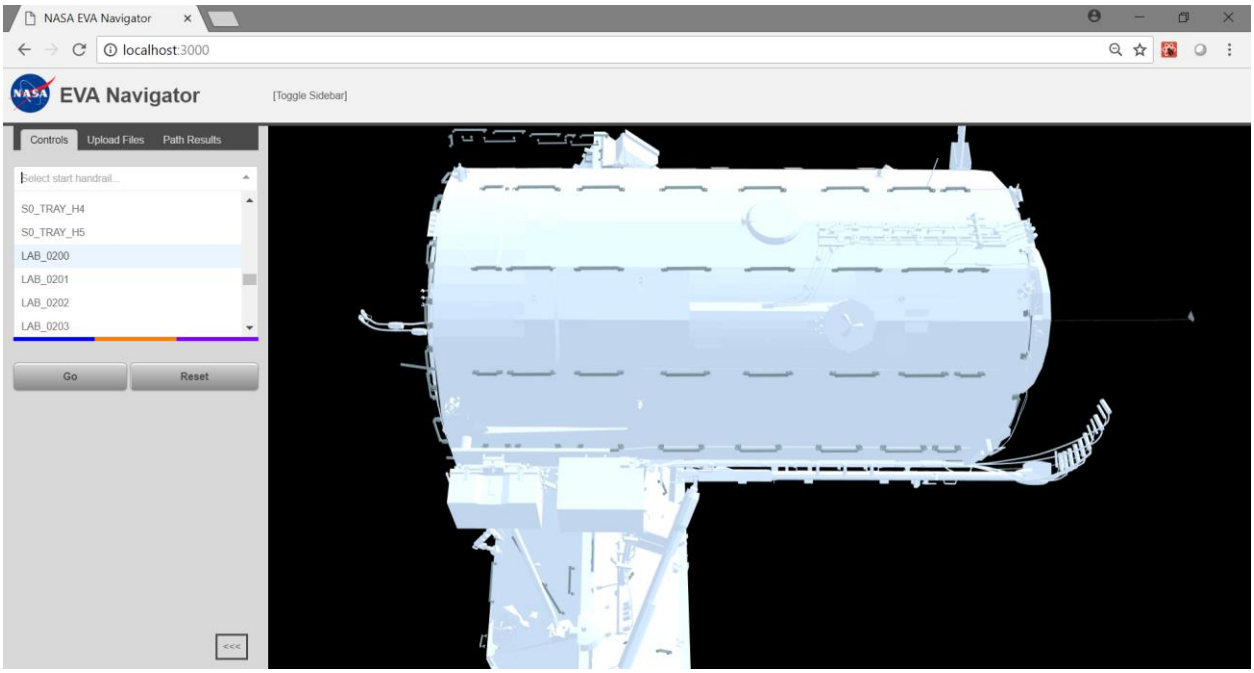

Figure 13. Test case 5 displaying selection of start handrail, LAB_0200.

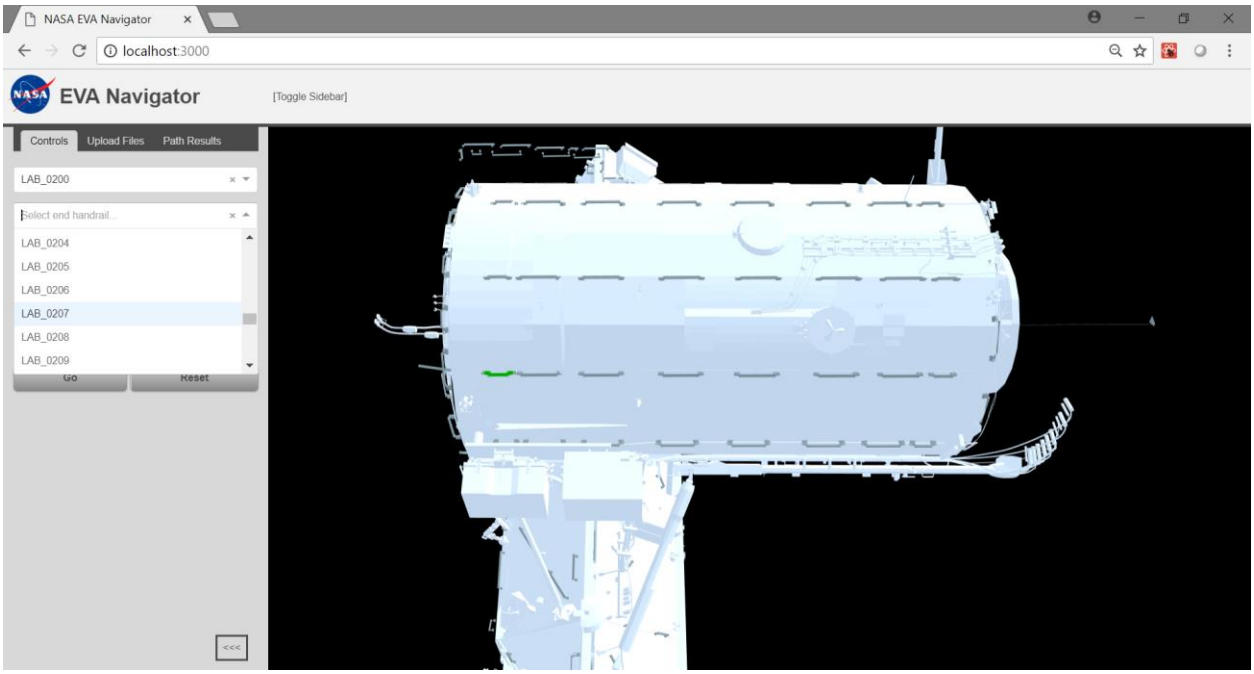

Figure 14. Test case 5 displaying selection of end handrail, LAB_0207.

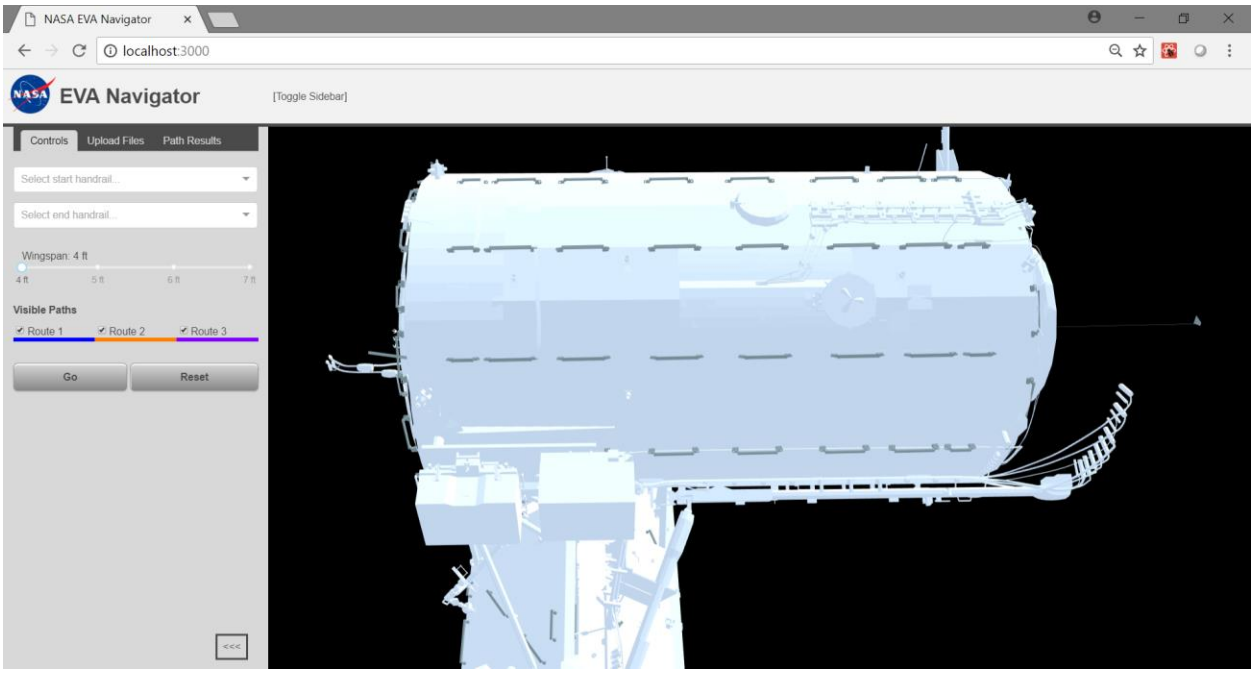

Figure 15. Test case 5 after deselecting chosen handrails, LAB_0200 and LAB_0207.

3.2 Assumptions and Constraints

3.2.1 Assumptions

It is assumed that the EVA Navigator web application has been setup and launched correctly, following the User_Manual.docx, section 3, Software Installation based on your operating system. It is also assumed that existing handrails in the ISS model properly correspond to the drop-down menus.

3.2.2 Constraints

Due to time limitations and lack of existing documentation on handrail mapping on the section of the ISS model used for functional testing, all handrails on the ISS model cannot be tested to ensure handrail names populated in the drop-down menus match the selected handrail in the model.

3.3 Findings

All functional tests worked as expected, resulting in the desired behavior.

References

None.