

A.W.A.R.E.S.

Advanced Wearable Augmented Reality EVA System

Austin Ebel (abe2122)
Asher Goldfinger (asg2189)
Roberto Interiano (rai2113)
Andrew Moshova (aom2115)
Daniel Nissenbaum (dan2135)

Columbia University
MEIEE4810: Intro to Human Spaceflight
Final Project: Milestone V
May 3, 2019

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 3 |
| 2 | Two Dimensional User Interface | 3 |
| 2.1 | 2D Overlay | 3 |
| 2.1.1 | Limiting Consumable and Time Remaining Component | 3 |
| 2.1.2 | Checklist Navigator Component | 3 |
| 2.1.3 | Suit Status Component | 4 |
| 2.1.4 | Display Detail Modes | 4 |
| 2.1.5 | Alerts | 6 |
| 2.2 | Physical Projection Placement | 9 |
| 2.3 | Estimated Time Remaining Calculation | 10 |
| 3 | Controls | 11 |
| 3.1 | Control Components | 11 |
| 3.1.1 | Mode Switch Component | 11 |
| 3.1.2 | Directional Control Component | 12 |
| 3.2 | System Integration | 12 |
| 3.2.1 | Display and Control Module | 12 |
| 3.2.2 | System Connection | 13 |
| 3.2.3 | Placement | 13 |
| 3.3 | Example Control System Procedure | 13 |
| 4 | Projection Systems | 14 |
| 4.1 | Image Transformation | 14 |
| 4.2 | Projection Module | 14 |
| 4.3 | Image Resolution and Field of View | 16 |
| 5 | Augmented Reality Demonstration | 16 |
| 5.1 | HoloLens and Augmented Reality Overview | 16 |
| 5.2 | Design Decisions | 17 |
| 5.2.1 | Mars Environment | 17 |
| 5.2.2 | Mars Rover | 17 |
| 5.2.3 | Textual Tutorials | 17 |
| 5.2.4 | Visual Tutorials | 19 |
| 5.2.5 | Voice Recognition | 21 |
| 5.2.6 | Integration with NASA | 22 |
| 6 | Conclusion | 22 |

| | |
|---|-----------|
| 7 Appendix | 23 |
| 7.1 Estimated Time Remaining Code | 23 |
| 7.2 JavaScript Animation Code | 24 |

1 Introduction

Extra Vehicular Activity (EVA) is often some of the most challenging and critical work done during any space mission. It requires hours of intense training, support from astronauts in the vehicle and from mission control, and an in depth knowledge of the tasks at hand. In addition, it is a difficult task, requiring immense physical and mental exertion on the part of the spacewalking astronaut. In the future, as we travel further from the support of Earth based mission control, it will become crucial for astronauts to be more self-reliant on EVAs, and to get the information they need quickly and efficiently, without the input of mission control.

Our solution to this problem is the Advanced Wearable Augmented Reality EVA System (A.W.A.R.E.S.). AWARES is a practical Heads Up Display (HUD) system projected onto the inner surface of the astronaut's visor that gives the space walker critical information regarding suit and wearer vitals, mission objectives, and technical information.

2 Two Dimensional User Interface

A.W.A.R.E.S. has two main elements to the User Interface: the 2D overlay and the 3D augmented reality environment. The 2D overlay displays textual information such as checklists and the system status. The 3D augmented reality environment displays three-dimensional objects and visual simulations that aid the astronaut in performing a wide variety of tasks. This section focuses on the various elements and design considerations of the two-dimensional interface.

2.1 2D Overlay

The 2D overlay is used to display varying levels of mission critical data pertinent to the astronaut, his spacesuit, and his mission. The UI elements appear in the corner of the astronauts vision and are transparent so as not to obstruct the astronaut's vision. There are four components that make up the 2D overlay. The first is a view shows time remaining on the mission as well as the limiting consumable. The second view shows various checklists that the astronaut can switch between. The third and fourth views show details about the spacesuit that are not critically pressing to the astronaut. To see a complete picture of our user interface in its most detailed states, see Figure 4 Below, we first discuss the four UI components in detail, and then discuss the various states of each component.

2.1.1 Limiting Consumable and Time Remaining Component

The most critical element of the user interface is the view that shows the amount of time the astronaut has before their limiting consumable runs out, calculated based on battery levels, carbon dioxide levels and oxygen levels. This information is displayed in the top-right corner of the astronauts field of view and is always on so that the astronaut is always aware how much time they have before they need to return from their spacewalk. The color of the text varies based on the amount of the most limiting consumable remaining and the amount of time remaining, as well as the presence of any errors or malfunctions. When there is more than 50% of the limiting consumable remaining, it will appear green, when there is less than 50% but more than 20% it will appear yellow, and when there is less than 20% remaining, the text will turn red. The text will also turn red if there is any pressing leak or malfunction of the delivery of any consumables. Simply, when text is red, the astronaut should be returning to the spaceship or base. Figure 1 shows the various possible color states of the component.

2.1.2 Checklist Navigator Component

In the top-left corner of the screen is the astronaut checklist navigator. This component provides a simple interface for the astronaut to view all his available checklists, select the list that he or she wants, and then

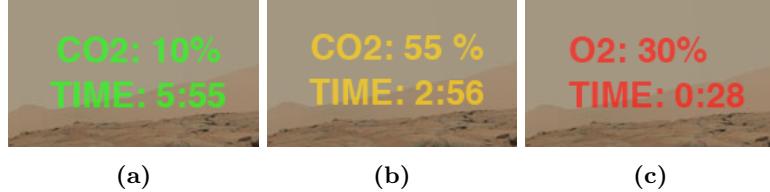


Figure 1: Various configurations of the limiting consumable time remaining component. (a) View is green there is an abundance of time/resources. (b) View is yellow when over 50% of limiting consumable has been used. (c) View is red when there is an emergency and less than 20% of limiting consumable is available or there is less than 30 min available.

interact with the checklist. The system will by default be populated with many of the pertinent checklists before the mission, but mission control could easily update the available checklists or send new ones as needed. The purpose of the checklist is to largely replace the physical checklist carried on the astronauts arm. The benefit of the digital checklist is that the physical one may become outdated and new procedures may need to be added, especially on long distance missions to Mars and beyond.

While more details about the physical mechanism used to control the checklist navigator are provided in a later section, the astronaut will generally use the checklist as follows. If the astronaut has a specific checklist in mind that he or she wants to access, they can either preload it before the mission or select it by voice recognition. They can also manually navigate to find a specific checklist. To find a checklist, the astronaut is first presented a list of categories that appropriately group the various checklists (Figure 2a). When they select one category, they are then shown a list of all the checklists in that respective category (Figure 2b). They can then select the specific checklist that they desire to view (Figure 2c). In the event of an emergency or a malfunction, this checklist view would automatically populate with the corresponding emergency checklist so that the astronaut has a quick actionable response to any problems that may occur.

The astronaut also has the ability to digitally interact with the various steps of the checklist. They can navigate through the steps, and “check” off the ones that they complete. The currently selected step is highlighted in blue and any completed steps are highlighted in green. Because the system is fully integrated with the spacesuit, the checklist will be smart enough to automatically “check” any items it can determine have already been completed.

2.1.3 Suit Status Component

The bottom left and right corners of the view show two components which update the astronaut on typically non-critical information about the suit. As pictured in Figure 3, the view is split up into two parts. On the left is the status of any variable that could be monitored by the astronaut but cannot be directly controlled (Figure 3a). On the right, is the status of any variable that can be directly controlled by the astronaut (Figure 3b). For instance, the *VOLUME* is directly adjusted by the astronaut on his or her suit.

While the astronaut will typically choose to not show this data as it is not critical, in the event of an emergency, the problematic data will automatically show. For example, if oxygen is running out, the oxygen status will automatically appear whether or not the astronaut has selected to display it.

2.1.4 Display Detail Modes

One of the major design considerations of the 2D overlay is to avoid distracting the astronaut by obstructing his or her view or providing him or her with too much unnecessary information. To address this issue, we will implement varying levels of detail which the astronaut can easily toggle among to clutter or unclutter the view with a physical knob. We distinguish between four different levels of details.

In the most detailed level, Mode 3, all non critical information, up to nine steps in the current checklist, and the complete suit status are shown. In the moderate detail level, Mode 2, only three steps of the current

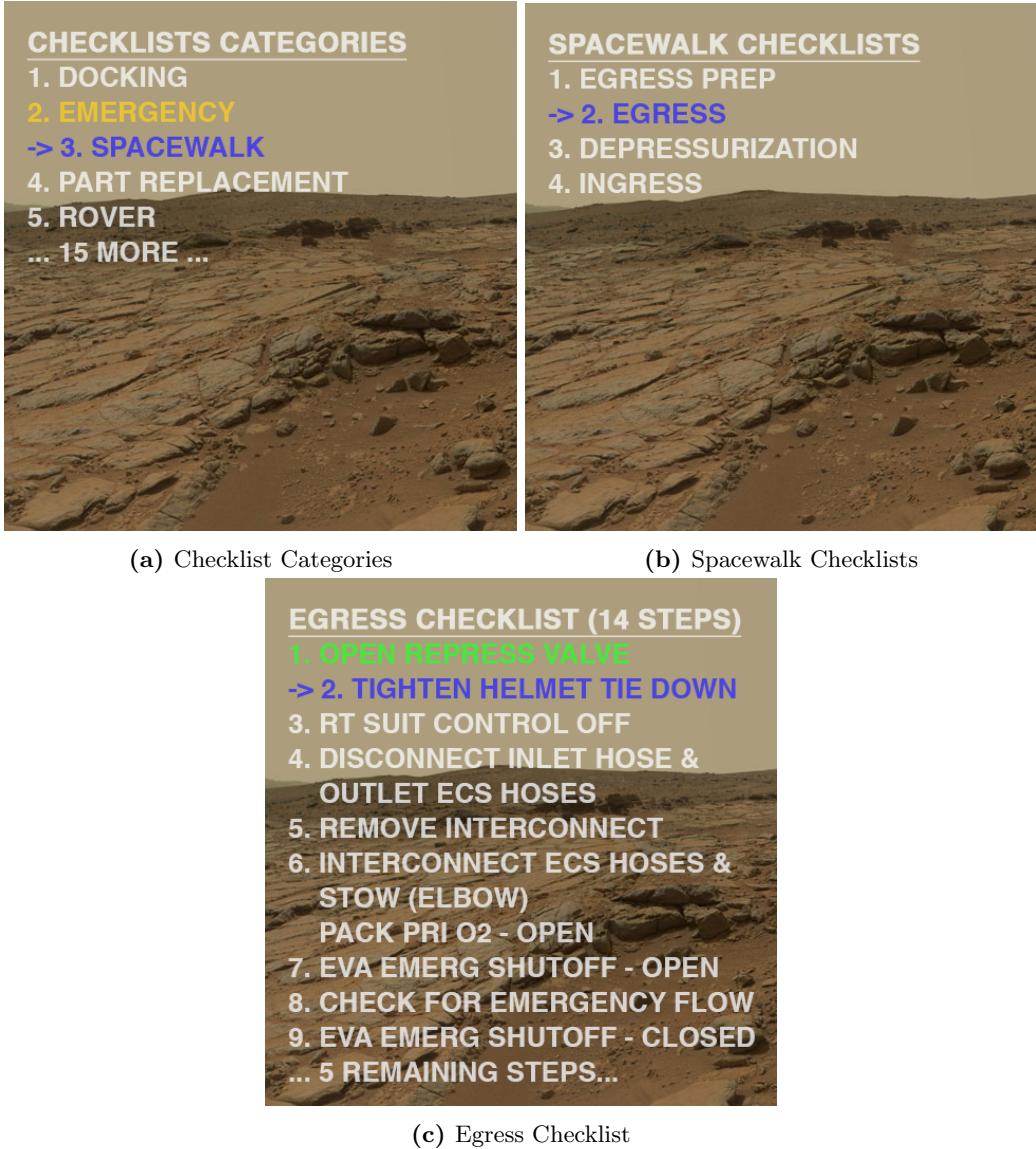
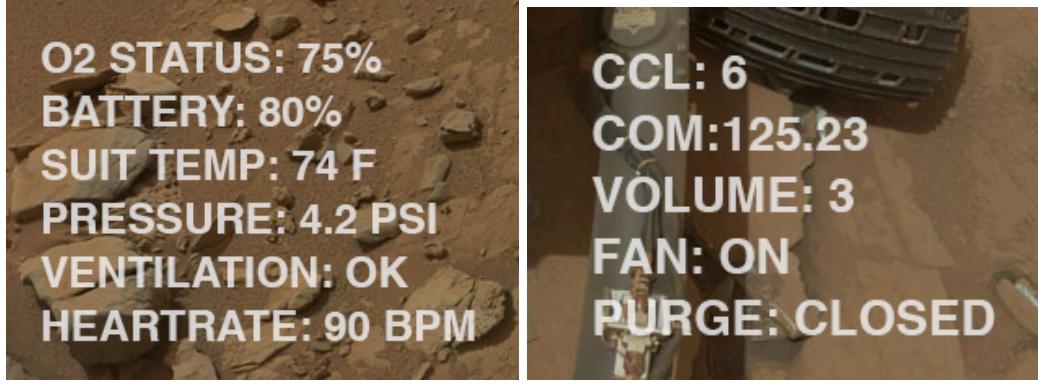


Figure 2: The Checklist Navigator Component and its various subviews. (a) Shows the initial view of the various categories of checklists. (b) Shows a single category or checklists. (c) Shows a single checklist with one completed step.

checklist will be displayed on the screen, and only the levels of the most limiting consumable and time remaining will be displayed. Non critical information will not be displayed in Mode 2. Mode 1, further reduces the clutter by only showing the currently selected step of the checklist. Finally, in the simplest mode, Mode 0, nothing will be displayed on the 2D overlay except the time remaining in the spacewalk.

We also include an additional emergency display mode. In the event of an emergency, a banner appears at the top displaying the current error message. The suit will automatically show the astronauts all the relevant details to diagnosing and addressing the issue and the appropriate emergency checklist is also displayed automatically.



(a) Left: Suit status

(b) Right: Suit control state

Figure 3: The left and right views of the suit status component (a) The left view shows the status of suit variables to be monitored. (b) The right view shows the current state of the suit controls.

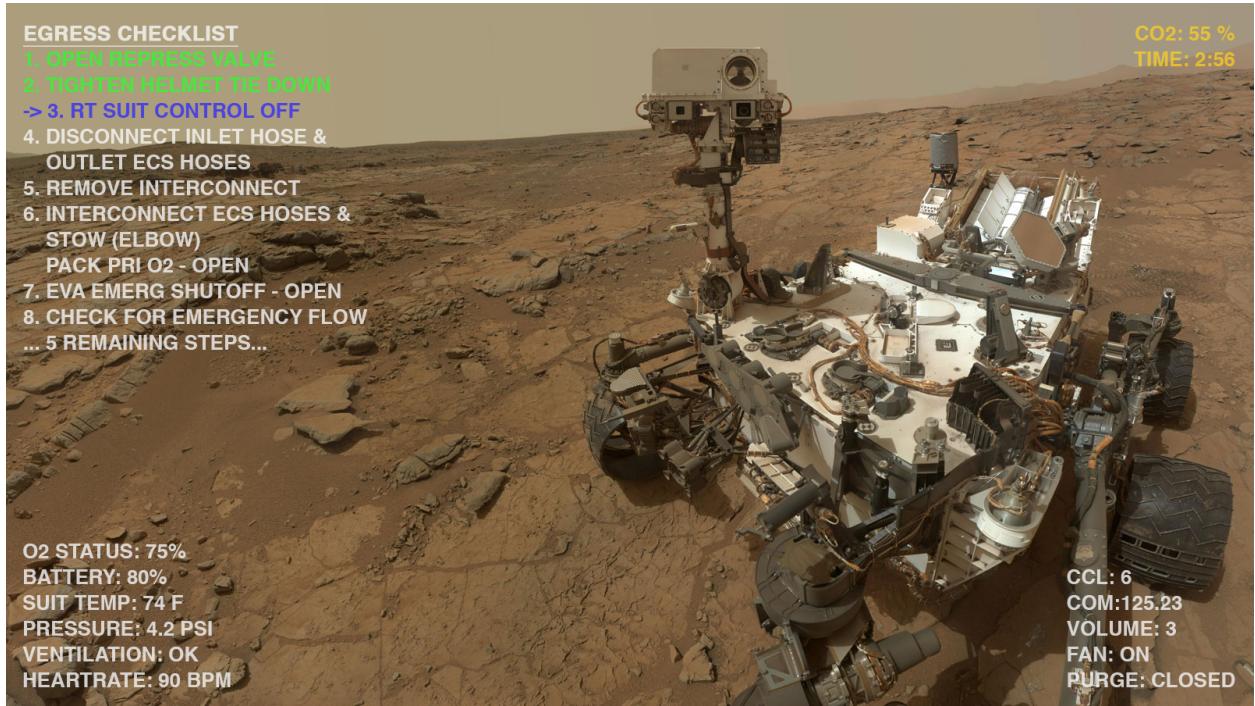


Figure 4: Mode 3: This is the 2D Overlay in its most detailed state.

2.1.5 Alerts

We also provide an interface for a wider variety of possible alerts. For instance, if the astronaut toggles a switch on his or her suit, a blue alert appears at the top of the screen to notify the astronaut of the change and new current value. These alerts can help keep an astronaut aware of his suit status and avoid inadvertently changing any settings. The alerts appear for 5 seconds and then disappear. An example of such an alert is shown in Figure 8

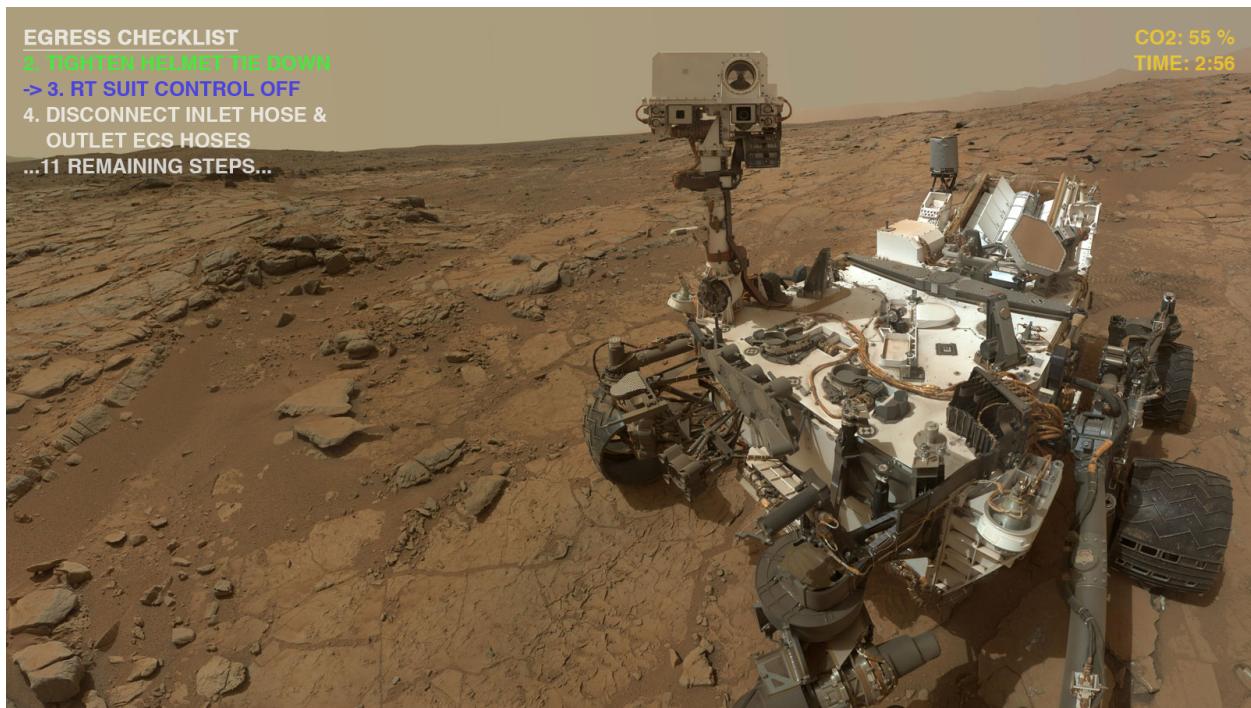


Figure 5: Mode 2: In this mode, the Suit Status Component is not displayed and fewer steps in the checklist are shown.

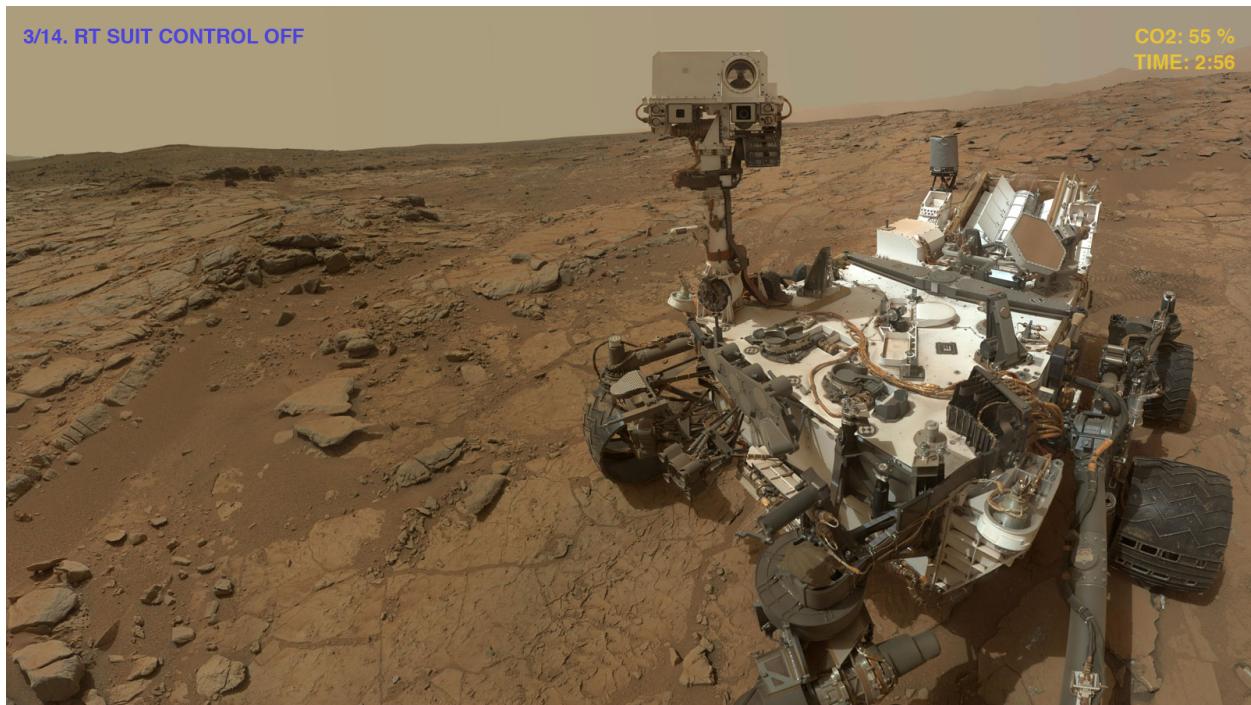


Figure 6: Mode 1: In this mode, the 2D Overlay only shows one step of the current checklist.

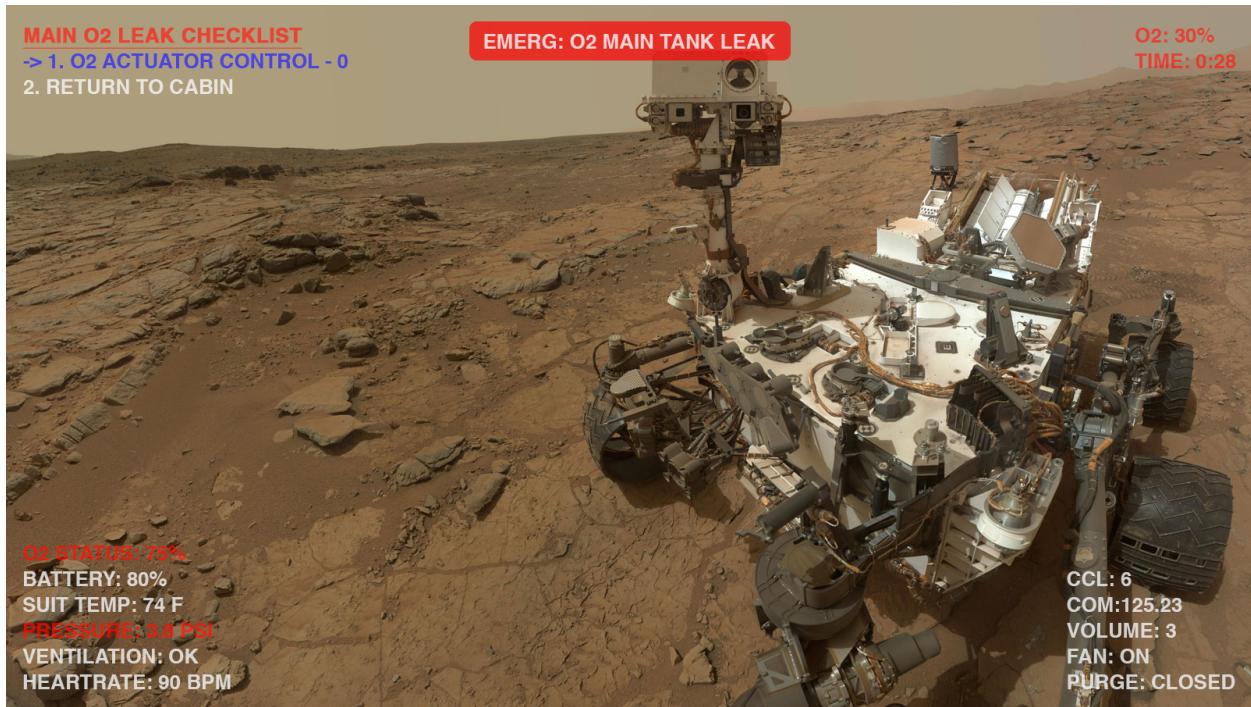


Figure 7: An example of the Emergency mode when activated.

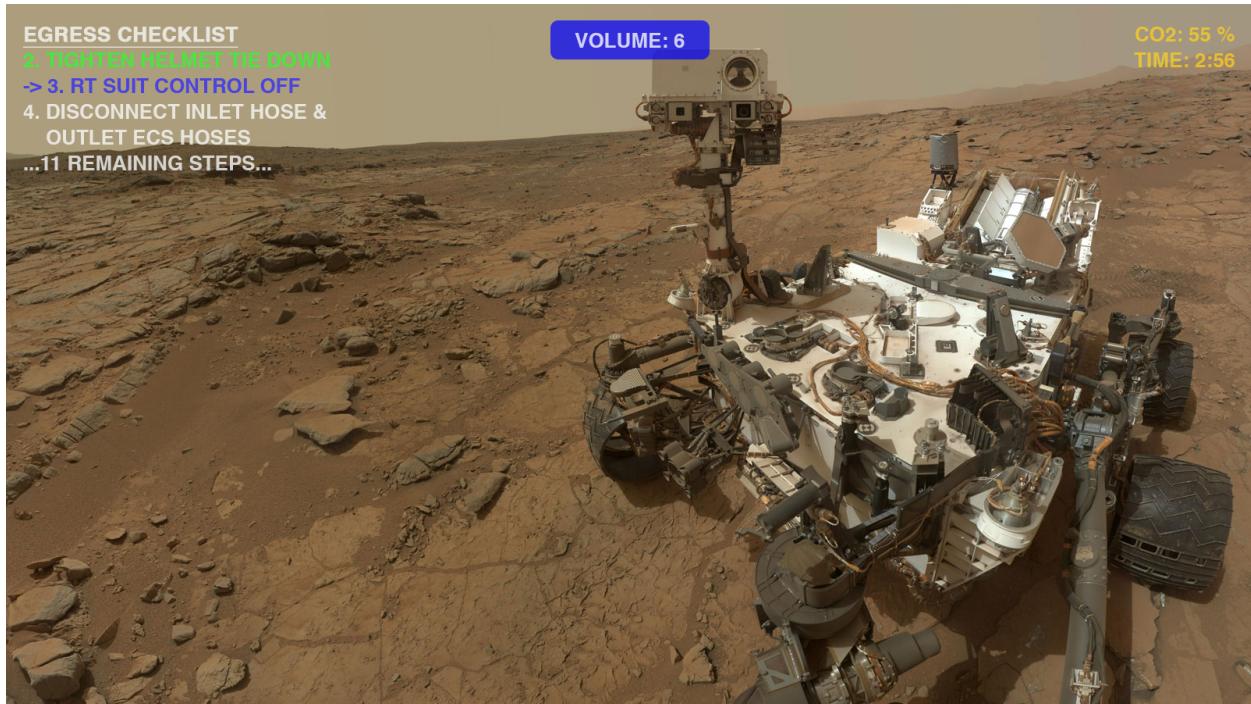


Figure 8: When the astronaut changes the volume on his suit, he or she receives an alert at the top of his view.

2.2 Physical Projection Placement

The decision on font size and placement of different UI elements was made based on two main factors: the physics of human peripheral vision and the limitations of the system's projection method. While the exact dimensions of NASA's spacesuit helmet are not public information, based on measurements of human heads and pictures on the internet it can be estimated that the helmet is roughly a 12 inch sphere with the eye-line about 4 inches from the front of the visor. The projection from eye-line level is capable of projecting a 90° horizontal field of view.

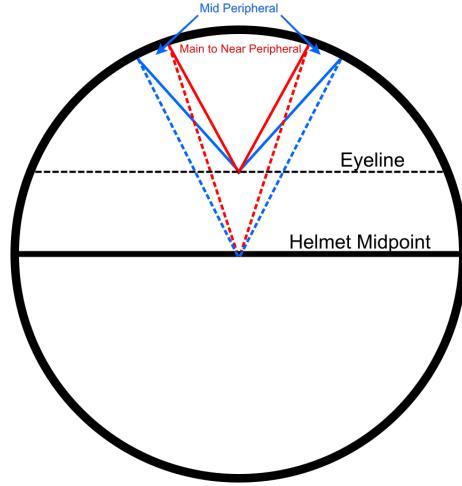


Figure 9: Visualization of the area of projection

Calculating the arc length of this full field first requires solving for the equivalent angle from the midpoint line. Two sides and one angle of a triangle that includes the solution angle are present: the radius of the full sphere is 6 inches, the distance from the midpoint and eye-line is 2 inches, and the largest angle in the triangle is 45°+90°, or 135° (as shown in blue in figure 15). Plugging these values into the law of sines equation results in:

$$180^\circ - 135^\circ - \arcsin\left(\frac{2}{6} * \sin(135^\circ)\right) = 31.37^\circ$$

Doubling this angle to get the full arc length:

$$\frac{62.73}{360} * 2\pi * 6 = 6.57 \text{ in}$$

As shown in figure 10, a human's mid peripheral vision begins at a field of view of 60°. In an effort to diminish and interfere with the user's central vision, all non emergency notifications are kept with this 60°-90°. Using the same method as before to get the minimum angle for this field of view, shown as the red triangle in figure 15:

$$180^\circ - 150^\circ - \arcsin\left(\frac{2}{6} * \sin(150^\circ)\right) = 20.41^\circ$$

$$40.81/360 * 2 * \pi * 6 = 4.27 \text{ in}$$

These results give 35.05% of the horizontal real estate to the side notifications, or 17.47% on each side.

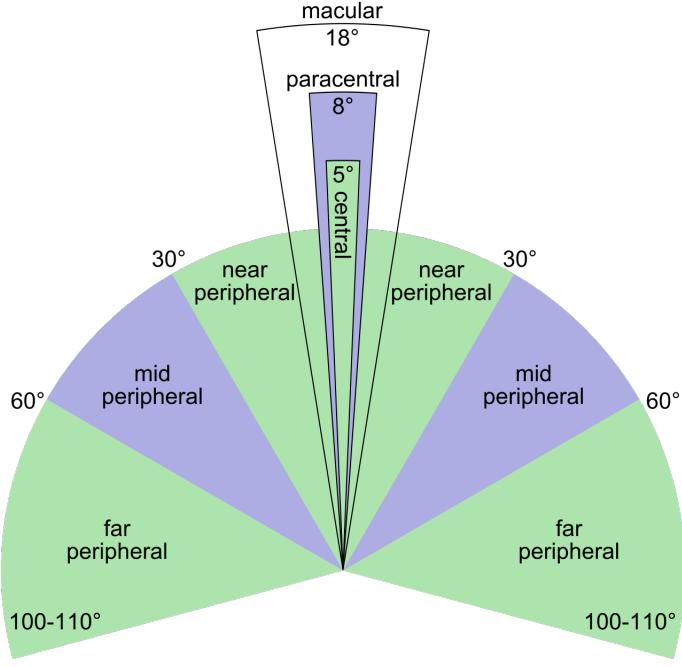


Figure 10: Peripheral vision of the human eye[8]

2.3 Estimated Time Remaining Calculation

The estimated time remaining is the only UI element that is present for all of the clutter modes of the system. It is meant to be similar to a fuel gauge in a car to provide the user with a constant reference point for how long they can stay out on their EVA, easing mental strain for the astronaut so that they are not forced to continually estimate themselves.

There are three main limiting consumables for an astronaut on their EVA: Oxygen levels, Battery capacity, and CO₂ filter usage. Our algorithm takes the rate of consumption of each of these consumables, calculates which one has the lowest estimated time remaining, and displays that time to the user.

Our original intention was to create a machine learning based algorithm to take even more factors into consideration, such as effects of heart rate or distance travelled on consumption, in an effort to create an even more accurate estimate. However after researching and experimenting with different methods for creating both a script and a training data set, we came to the conclusion that adapting the benefits of machine learning makes very little sense for our application, at least for the foreseeable future. Machine learning works by building and adapting a model that is trained using a very large set of data. The more data used to train, the more accurate the system becomes. For a Bayesian network[5], the most likely choice for a model doing a time estimate from many variables, a small data set would likely result in connections being made that are not actual correlations, and could cause incorrect estimations.

Since AWARES is targeted at EVAs on Mars and beyond, there will be a lack of reliable and actionable data that we can use to train a successful model until there have been countless EVAs in similar environments. However the estimated time remaining should not be as simple as just taking the consumption rates of all three limiting factors as usage rates can vary drastically in a small period of time. Instead, our algorithm takes a weighted moving average of each factor, using the average rates over the last minute and assigning more weight to the more recent data points to get a more accurate estimate. We then take the remaining amount of each consumable and multiply it by the calculated average rate. In addition we want our algorithm to learn from the user's past EVAs, so it also takes into account time and consumable usage of every past mission. The more EVA data the astronaut has, the more their past usage will be weighted into the calculation. Starting at 5% weight and increasing by 5% with each new data set (at a max of 50%), we combine the current rate data from past rate data to get our final estimate, displaying the minimum to the user.

Extraneous situations or emergencies during an EVA can alter the rates drastically, so anytime an emergency is present with one of the three systems the algorithm no longer takes last data into consideration and the moving average weighs more recent data points even more. The moving average equation is:

$$Avg = \frac{\sum_{t=1}^n W_t * Vt}{n} \text{ where } \sum_{t=1}^n W_t = 1$$

W represents the weight assigned to each value, and V represents the consumption value itself. The weights increase with t, all add up 1, and end with a weight of 10%. An example implementation of the code is given in the appendix 8.1.

3 Controls

User navigation of the A.W.A.R.E.S Interface will be accessible by way of voice control and a physical control system which will be integrated into the Extravehicular Activity Mobility Unit (EMU). The control system will enable the astronaut to manually navigate the options available on the 2D overlay display. This section focuses on the components, system integration, and design considerations of the A.W.A.R.E.S control system.

3.1 Control Components

The physical control system will consist of two parts; the mode switch and the directional control component. Each component will have a unique functionality providing effective navigation of the 2D interface.

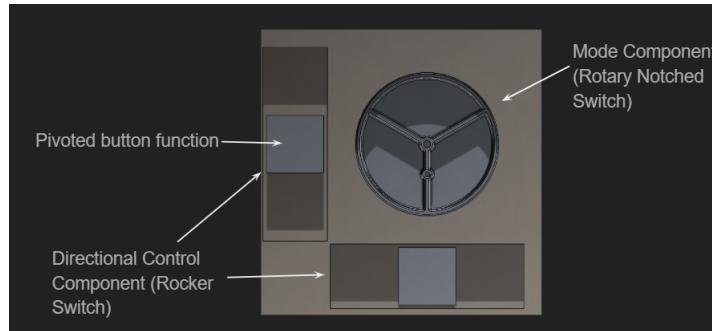


Figure 11: A.W.A.R.E.S Physical Control System

3.1.1 Mode Switch Component

The mode switch will consist of a rotary notched switch allowing the astronaut to choose from the four levels of detail, or Modes, (Section 3.1.4). The switch will only be capable of moving 120 degrees notched at four locations in order to dictate the Mode selected by the astronaut. The internal design of the mode component is modeled off the Cooling Control Valve which is currently centered at the left end of the Display and Control Module. The mode switch will be printed so that the actual handles are almost flush with the outside, however, the plate of the switch will be half an inch deeper into the surface allowing for protection against accidental movements of the switch.

3.1.2 Directional Control Component

The second component of the control system is the directional system component which will allow the astronaut to navigate the options presented on the selected Mode. This system will consist of two rocker switches, one oriented horizontally, and one oriented vertically. A rocker switch functions by “rocking” in one direction to break the circuit and the other direction to connect the circuit. The actuator allows the switch to rotate back and forth, and the status of contact determines whether the switch is set on or off. As the switch is moved from its centered location it connects the circuit, and, therefore, making the commanded selection. Holding down the switch will move 1 item per second for the first 3 seconds of holding, 2 items per second for the next 3 seconds of holding, and 4 items per second for all remaining seconds of holding. The rocker switches will have a width slightly larger than that of an astronaut glove finger- 1.125 in.

The horizontal rocker switch will be restricted to a horizontal axis, and it will be centered, meaning that force is required to push it left or right. The purpose of the horizontal switch is for the astronaut to be able to select in, and out of different categories, and checklists. It will basically serve as a select tool, and a “go back” tool. The switch will function under the field of view of the astronaut, which means that a movement of the switch to the astronaut’s right, will select an item. A movement of the switch to their left side will then take the astronaut back a step.

The vertical switch will be restricted by vertical single axis movement, and will be centered so that force is required to push up or down. This switch will be used for scrolling through tasks within a checklist. A unique function of the vertically oriented switch is its button functionality. On top of a rocker, the switch will also serve as a button, so that the astronaut can highlight, or “check off” completed tasks. In order to facilitate this, the top surface of the switch will have a filleted surface so that the astronauts glove finger can easily press down when necessary. In order to avoid accidental presses of the button while moving the vertical switch, the button will have a pivot point.

Since the vertical switch only functions within a checklist, when the display is set in modes 0 and 1, nothing will happen when the rocker switches are moved or the button on the vertical switch is pressed. In modes 2 and 3, the list of checklists will be present in the top left hand corner of the field of view and the switches will be used to navigate the checklists.

3.2 System Integration

The control system will be located in the Display and Control Module (DCM) which basically serves as the control panel for the EMU.

3.2.1 Display and Control Module

The control system will be located in the Display and Control Module (DCM) which basically serves as the control panel for the EMU. The DCM contains a series of mechanical and electrical controls, a microprocessor, and an LED display. The DCM also enables the crew to control the Primary Life Support System and the second oxygen pack. Internal connections are routed to the EVA communicator through the EVA electrical harness, cable routes signals from electrocardiogram sensors, as well as caution and warning signals to crew members headset.

The DCM is connected to upper torso through internal and external hookups. Multiple function connector links the display module to the service and cooling umbilical enabling controls during suit checkout. Sensors in the PLSS supply information to the display and control module where the microprocessor maintains an automatic watch over system integrity. The DCM interacts with a microprocessor in the PLSS that contains a program that enables the crew to cycle the display through a series of system checks and determine the conditions of a variety of components. The microprocessor allows the crew to monitor oxygen pressure, alarm at high oxygen intake in primary tank, monitors water pressure and temperature in cooling garment, and alarm at high carbon dioxide. Furthermore, the DCM has a fiber optic cable used when MMU (man maneuvering unit) is connected to the EMU. When connected it provides cycled readouts of propellant

pressures, temperatures, and battery conditions.

3.2.2 System Connection

The control system will be connected to the microprocessor in order to provide the monitored information that is to be displayed at the corresponding suit status levels. Currently the wrist mirror serves as the primary method of observing the control status. Displaying this information in the suit status levels will facilitate status updates. installing the system to the currently established connections in the Display and Control Module is the most efficient method of integration.

3.2.3 Placement

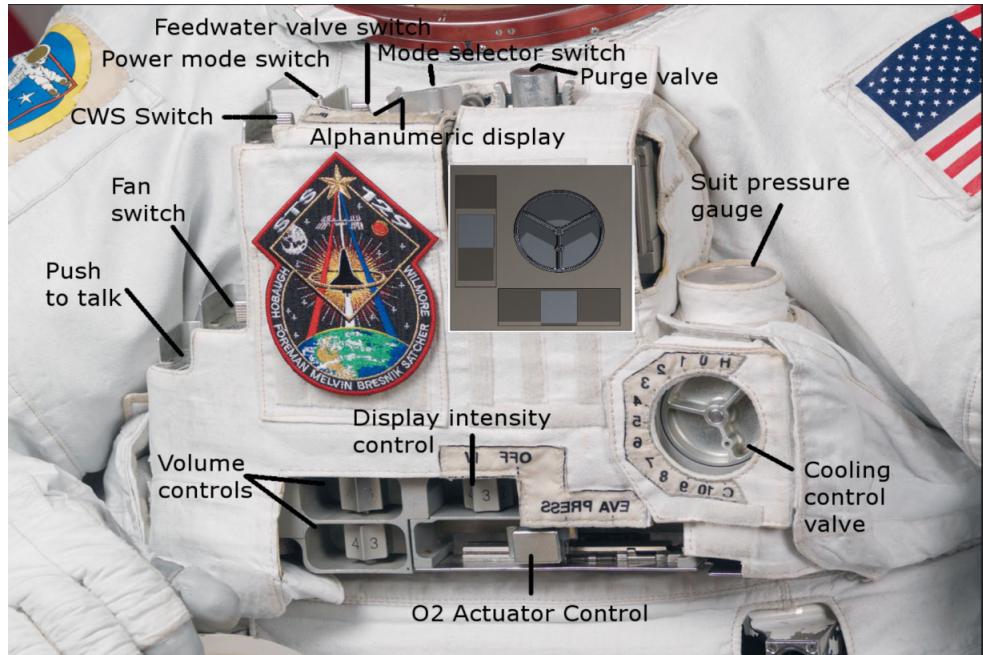


Figure 12: A.W.A.R.E.S Physical Control System placement in Display & Control Module

As illustrated in Figure 11, the mode switch will be centered at the currently-available upper portion of the Display and Control Module, under the purge valve. This was decided because of its distance from the cooling control valve which has a very similar design, and therefore, might cause confusion if situated on the same plane of vision.

3.3 Example Control System Procedure

Mode 2 is selected on the display. The top left of the screen will show a list of a few available checklists. Move the horizontal rocker switch to the right in order to select into a category of checklist. Once in the chosen category, move the horizontal rocker switch to the right once more to select a specific checklist. The view will now show the steps within the selected list. When a step has been completed, press the button on the vertical rocker to “check it off” and move to the next step of the checklist. To go back to the full list of checklists, move the horizontal rocker to the left. If at any point you want to take make all the checklists disappear, just move the mode switch to either Mode 0 or Mode 1.

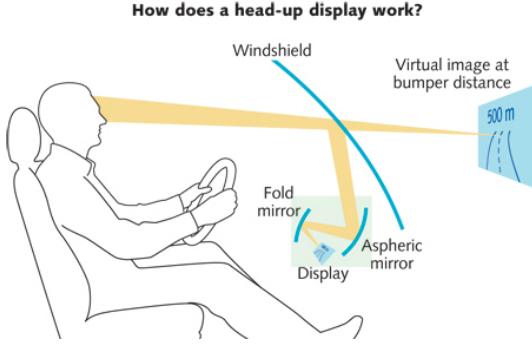


Figure 13: Standard HUD projection optics.

4 Projection Systems

The augmented reality interfaced described in this paper would be implemented by a multi-part projection system mounted on the interior of the EMU helmet. The image would be projected onto the helmet visor, but would appear to be at a distance of 10-15ft. This significantly reduces eye strain on the user of the HUD, as it doesn't require focusing alternatively on the nearby image on the visor and the distant image of the world.

4.1 Image Transformation

Due to the geometry of the possible locations for the projection modules, and the roughly spherical shape of the visor, it is impossible to form a virtual image at a distance from the surface of projection, as is common in classical HUD systems.

Generally, the image is reflected off an effective mirror which runs at a 45 deg angle to the horizontal line of sight of the viewer. However, with a spherical mirror analogue with the viewer roughly at the center, as in an EMU helmet, such an image can only be created if the source is located at the center of the sphere - a location inhabited already by the viewer's head.

To circumvent this problem, the required image can be simply projected directly onto the inner surface of the EMU visor. The difficulty then becomes making the image appear at a distance from the surface of the visor. This can be achieved by processing the image to project the intensity field that would appear on the surface of the visor if the required image were to exist at the required distance. This is a complex and computationally expensive problem to solve using brute force methods, but it can be solved using an understanding of the optics of the system.

Computing this image efficiently falls under the purview of Near to Far Field Transformation, a technique used frequently in the design of antennas and radio optics. The technique involves taking the 2D Fourier Transform of the image at the distance, and using it to calculate the Fourier Transform of the resultant image on the surface of the visor. The reverse Fourier Transform can then be calculated to determine the image that needs to be projected on the visor.

4.2 Projection Module

Projection is accomplished through the use of two small projectors, each mounted on either side of the astronaut's head in the helmet. The average Bitragon breadth measured from one ear base to the other is 5.71". Approximating the EMU helmet as a sphere of diameter 12", this leaves 3.1" for the projection system to fit into on either side. Each projector unit is roughly 1"x1"x3", only taking up 33% of the space on either side of the head in the helmet and providing enough remaining clearance for astronauts to comfortably move their head.

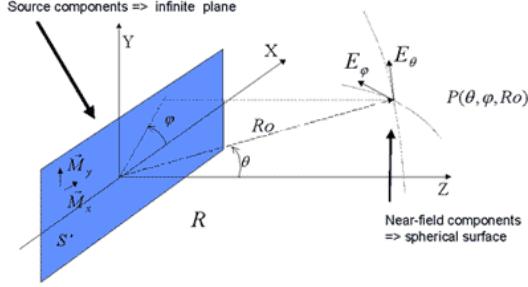


Figure 14: Near Field to Far Field Transformation

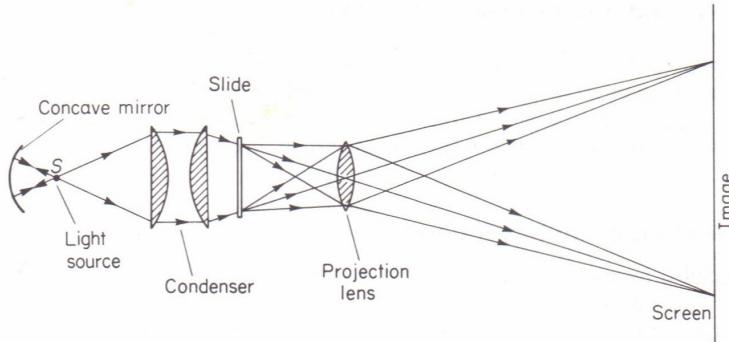


Figure 15: Classical Projection Optics

A typical projection system consists of a light source, an image to project, and a lens, as shown in Figure 15. The light source for this unit will be a high powered, efficient LED, such as the Cree CXA 1310 High Density LED Array, using only 1W of power, but providing enough 1445 lumens of light, enough to project onto a transparent surface at the distance required. The image will be provided by a HED 2200 Color LCOS Microdisplay, which provides 1280x720 resolution, while having a size of only 5.83 x 3.24 mm. The associated microdisplay electronics can be packaged in the rear of the device.

This projector is unusual however, in that it must project onto a relatively close and strongly curved surface, necessitating a large depth of field, or the distance from the lens in which the image is in acceptable focus. To achieve this, an aperture is added to the projector, slightly limiting the brightness of the image produced by the projector, but dramatically increasing the depth of field. The depth of field is given by the following equations, where H is the hyperfocal distance, f is the lens focal length, s is the focus distance, D_n is the near distance for acceptable sharpness, D_f is the far distance for acceptable sharpness, N is the f-number, and c is the circle of confusion.

$$H = \frac{f^2}{Nc} + f$$

$$D_n = \frac{s(H - f)}{H + s - 2f}$$

$$D_f = \frac{s(H - f)}{H - s}$$

The distance to the center of the image projected by each unit is roughly 6.7". With a lens with an extremely short focal length of 0.23" and an f-number of 1.05, an image can be projected on the visor and be acceptably focused for a range of around 2", allowing for an image to be projected for around 45 deg in either direction from the center of projection for each unit, with a 10% loss in total image brightness. As technology improves, and microdisplays shrink further, this can be improved, until an additional aperture is not needed, and no light is wasted.

4.3 Image Resolution and Field of View

When combining the two projectors, a field of view of 80 deg vertical and 90 deg horizontal can be obtained, covering the area of maximum focus of the human eye. The total resolution of the image created will be 1280x1440 pixels, covering a total area of 79in^2 . The average resolution of 153 pixels per inch, providing a good quality picture to the user. This too, will be improved upon in the near future, as microdisplays achieve higher resolutions in the same limited space.

5 Augmented Reality Demonstration

In addition to the 2D user interface and overlay, we created an augmented reality application using Microsoft’s HoloLens to demonstrate the full functionality of AWARES. The goal of this application is to provide tangible evidence for the necessity of AWARES, motivate its development in the first place, and display its applicability to NASA. For reasons discussed in the following sections, our demonstration involves a total augmented reality experience centered around fixing a rover tire on Mars. A link to our final demonstration can be found here: <https://www.youtube.com/watch?v=bCG6xDrOKCA>.

5.1 HoloLens and Augmented Reality Overview

As mentioned previously, we created our augmented reality experience with Microsoft’s HoloLens. This is primarily due to the fact that it was readily available to us through Columbia’s Mechanical Engineering department, however it is also the most supported augmented reality tool currently available, making it ideal for both current and future development.

There are several software tools that integrate with the HoloLens. Throughout our development process, we tested two of them: Unity and Vuforia Studio [7, 9]. Unity is a more general game development platform, whereas Vuforia Studio is specifically targeted to HoloLens applications. Initial work was done in Unity due to its direct support from Microsoft through their Mixed Reality Toolkit. However Unity quickly became a less viable option as we found it was much less well-supported than we had initially thought. Thus, we moved to Vuforia. Vuforia specifically targets the HoloLens and provides a wide array of tools in their integrated development environment, Vuforia Studio. This made creating our demonstration significantly easier, and simplified programming to strictly animations.

Below is a brief description of the development procedure within Vuforia Studio.

1. *Import a CAD model* - For our demonstration, we used a CAD model available on NASA’s Jet Propulsion Laboratory’s GitHub page [6], originally created for a community project. Many smaller, negligible components were removed in SolidWorks to increase the framerate while using HoloLens. Separate work was done in SolidWorks to ensure we could animate the removal procedures correctly, thus multiple CAD models (base rover, wheels, and individual bolts) were imported separately.
2. *Import 3D Images* - Vuforia Studio supports what’s known as ”3D Images”, or their idea of placeable 2D images within a 3D environment. We created our associated text tutorials through these 3D-Image components [9].
3. *Create animations on CAD models* - Animations were done with JavaScript and directly supported within Vuforia Studio. A more detailed look at the animation procedure can be found in the subsection *Visual Tutorials*.
4. *Assign animations to commands* - Vuforia Studio directly supports assigning animations to either voice commands or hand motions while wearing the HoloLens.
5. *Publish and load onto HoloLens* - Upon completion, the experience can be published, generating a scannable QR code. Downloading the app, Vuforia View, on the HoloLens allows the user to directly scan that QR code and download the experience to the HoloLens.

5.2 Design Decisions

Significant thought was put into crafting an ideal augmented reality experience to display the full capabilities of AWARES and the HoloLens. Below we will talk more about each design decision, as well as go in depth into what each component looks like in our project.

5.2.1 Mars Environment

We felt creating a demonstration around some situation on Mars was ideal for a few reasons. Primarily, we felt the true potential of AWARES is more evident when it is impossible to quickly communicate with Earth. On the ISS, for example, astronauts will have constant feedback, advice, and oversight from Mission Control. Thus an augmented reality tutorial, while still beneficial, is much less impactful to the success of an EVA. On Mars, however, fast and direct communications with Earth are an impossibility due to the roughly 40 minute round trip communication time. This highlights the need for complete and total preparation *before* starting an EVA, as communication is only possible with other astronauts on Mars.

Due to this, there must be some sort of more advanced checklist to accompany astronauts on EVAs on Mars. We feel that we can migrate this pre-existing necessity to an augmented reality application. Not only does this mask the issue of direct communication by providing astronauts with EVA specific tasks, it does so with audio, visual, and textual feedback. Thus, anything NASA feels is relevant to show astronauts can be uploaded directly to their augmented reality visor, ensuring astronauts are as confident and comfortable as possible during their EVAs.

5.2.2 Mars Rover

Naturally, after choosing a Mars environment, we felt performing maintenance on a rover was incredibly fitting. Not only does it give us a set end goal, but it does so with an experience that may ultimately become a reality. We felt a tutorial procedure for removing a tire on the rover was a fitting demonstration of the augmented reality experience, however the topics of future tutorials are limitless. We split the tutorial into 3 stages: removing the bolts on the wheel, removing the wheel itself, and removing the rim of the wheel. We felt this adequately displayed the usefulness of our textual and visual tutorials. Again, there is no limit to the complexity these tutorials can have. While it is important to keep them quick and easy to understand, NASA could include *any* information they desire and *any* visual they deem necessary.

As mentioned previously, we found a CAD model for the rover through a community project at JPL. While not incredibly close to the true rover design, we felt it was fitting enough to work with in our project. Below, we will talk about each component added to our augmented reality demonstration, and our reasoning behind including it.

5.2.3 Textual Tutorials

Textual tutorials integrated into the environment were designed to provide astronauts with a more detailed checklist of each step during an EVA specific task. These text tutorials were aimed to mimic a traditional checklist astronauts would have while doing an EVA. Thus, they provide additional information about each step in the tutorial procedure, and highlight key considerations astronauts need to know about. Examples of our text tutorials can be seen in Figure 16. In addition, Figure 17 shows the placement of these text tutorials in the 3D environment.

Several design considerations went into the making of these text tutorials. For example, the content of each slide is obviously very important, however the placement of the slide relative to the rover component in question, the opacity of each slide, and the size of each slide are also incredibly important. Furthermore, through our testing, we found a static tutorial slide placed in the environment was harder to work with than one that rotated with the user. This rotation of elements is known as *billboarding*, and can be seen fully

Left Front Wheel Disassembly Tutorial

(RLWF – Procedure 1.0.3)



Note: Information in the bottom panel is purely supplementary. It is visible because you have selected "HUD Setting 3" for most information visible. Information in the lower panel will walk you through each step of the tutorial process more carefully. If you have done this before, or feel confident, say "Remove Lower Panel" to disable this functionality. Finally, for this tutorial, we have not shown the process of placing the rover on your associated jack stands.

You have selected the Front Left Wheel Disassembly Tutorial for the Curiosity Rover. In here, you will:

- Remove 8 bolts securing the wheel.
- Remove the tire from the wheel assembly.
- Remove the rim from the tire.

If this is the correct tutorial, say "Next" or navigate to the associated switches on your suit.

(a) Introduction

Step 2: Tire Removal

Slide the tire off the wheel assembly structure.



(Seen in the animation for this stage.)

Note: Say "Replay" to replay the current tutorial step, or navigate to the associated switches on your suit. The jack stand is not shown in the animation.

Follow the tire removal procedure below:

- Grip both sides of the wheel, and mimic a steering motion left and right while sliding the wheel off the mount.
 - With the bolts removed, this should be an easy process.
- Place the tire on the ground.

Once completed, say "Next" to advance to the Rim Removal Procedure.

(c) Tire Removal

Step 1: Bolt Removal



Remove bolts:
RLF1.04 – RLF1.09

(Seen in the animation for this stage.)

Note: Say "Replay" to replay the current tutorial step, or navigate to the associated switches on your suit.

There are **6** bolts that need to be removed in order to free the tire:

(RLF1.04, RLF1.05, RLF1.06, RLF1.07,
FLWF1.08, and RLWF1.09)

Use your socket wrench to first loosen each of the 6 bolts. Ensure proper retrieval procedures below:

- Retrieve accompanying storage bag from the left back pocket of your toolkit.
- Remove each bolt, and place in bag.
- Return bag to left back pocket of toolkit.

Once completed, say "Next" to advance to the Tire Removal Procedure.

(b) Bolt Removal

Step 3: Rim Removal



Remove the rim with a crowbar.

- Monitor vital signs carefully.

(Seen in the animation for this stage.)

Note: Say "Replay" to replay the current tutorial step, or navigate to the associated switches on your suit. This is a very labor intensive process. Proceed carefully and always monitor your vital signs!

Follow the rim removal procedure below:

- Place the wheel flat on the ground.
- Remove the crowbar found in the right back pocket of your toolkit.
- Take the crowbar and firmly pry the wire mesh off the rim in a circular motion
 - Rover tires have different removal procedures to normal tires.
- Once removed, return the crowbar to the right back pocket of your toolkit.

Once completed, say "Next" to complete the tutorial. If you plan on replacing the wheel, say, "Replace Left Front Wheel", to continue to the correct replacement tutorial.

(d) Rim Removal

Figure 16: Content of our tutorial slides

implemented in our final video demonstration. The remaining information in the section will deal precisely with each aspect mentioned above, and motivate its implementation in the first place.

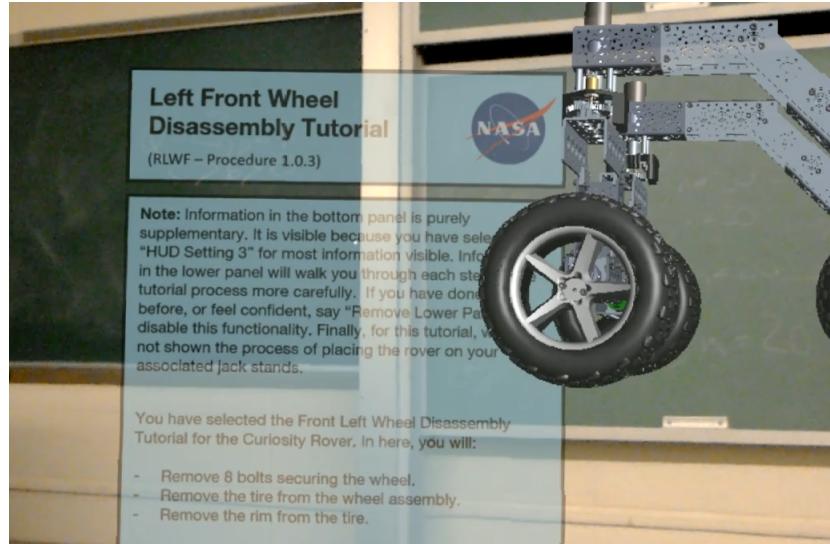


Figure 17: Tutorial slides in 3D environment

As seen in Figure 16, each tutorial slide has been separated into two components. The top component includes critical information to the success of each tutorial stage. The bottom panel is used for more supplementary information, and a more detailed walkthrough of each stage at hand. Ideally, this functionality would be integrated into the mode selection of the 2D user interface. Choosing Mode 3 would provide the most detailed information about each stage, while choosing Mode 0 would only provide the top panel, and its most critical information. Furthermore, the style of writing was aimed to mimic NASA's own procedural style of writing. This means writing quick, bulleted procedures, that are easy to read and follow. In addition, information required before the start of each tutorial stage is put in a "Note" section at the top of the supplementary panel - again, following NASA procedures.

As for the placement of each tutorial stage, through a trial and error process, we found placing them at roughly a 15° to the left of the rover while standing around 4 feet away felt like an optimal position. This meant each tutorial stage would be out of way of any rover animations, however not far enough so as to require astronauts to significantly move their head to read them. Furthermore, the size of each tutorial stage was incredibly important to determine. The limited field of view of the HoloLens meant that slides too large would be cut off. Slides too small, on the other hand, would be too hard to read. We settled on the size seen in Figure 17, as we felt that was an ideal combination of the two considerations above.

Billboarding was employed to ensure each tutorial stage always faced the astronaut. Thus, no matter the location of the astronaut in the environment, each 2D tutorial will always be directly facing them.

Each text slide was made to ensure it followed the visual component of each tutorial stage (more on the visual components in the following section). This meant assigning one slide to one tutorial animation. In the future, multiple animations could be assigned to one tutorial stage, fully detailing every aspect of even the supplementary page as well. All in all, we felt purely visual tutorials left much to be desired, and limited the scope of a project like AWARES. Implementing 2D textual tutorials removes the necessity for a separate piece of technology to keep track of checklist materials, and ensures astronauts can always reference each subsection of the tutorial stage with minimal effort.

5.2.4 Visual Tutorials

At the heart of AWARES and our augmented reality demonstration is the use of visual animations to aid astronauts in the tutorial process. This means animating each stage within the process to provide critical information not attainable through current NASA implementations and without the use of augmented reality. Again, by including these visual tutorials, we hope to reduce the stress of EVAs that are not monitored quickly

from Mission Control, and increase the confidence of astronauts throughout their EVA process. As seen in our final demonstration video, animations for each tutorial stage were played using a voice command, and complement the associated text tutorial.

Our demonstration consists of 4 animations: lifting the rover up to eye-level, removing 8 bolts in the left front wheel, separating the tire from the rover's wheel assembly, and removing rim from the wheel. Figures 18-21 show the modifications to our rover after each of these tutorial steps.

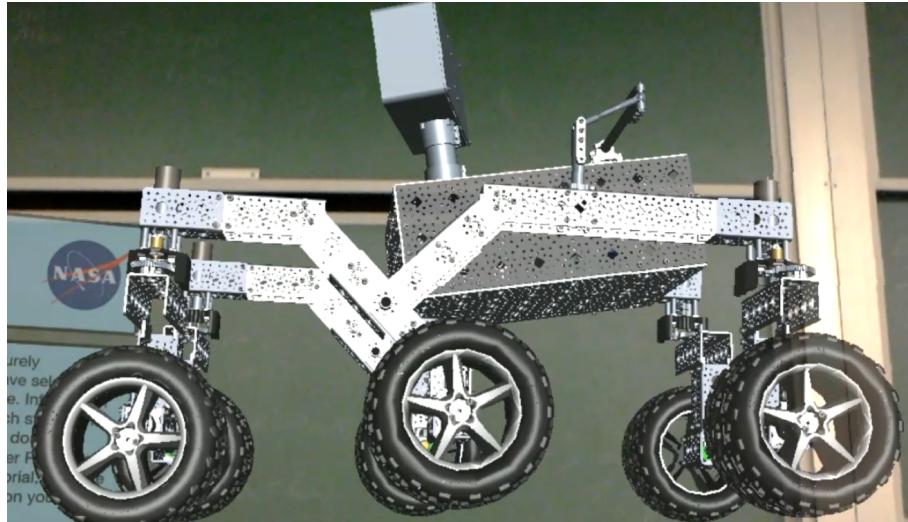


Figure 18: Moving the rover within our environment to eye level



Figure 19: Visualizing the bolt removal process

Animations were done directly inside of Vuforia Studio using JavaScript, and mapped to voice commands and procedures. Appendix 8.2 contains the JavaScript code required for animating a single object within Vuforia Studio. This was done for all 11 animated objects. Animations were created by updating the position of the animated object by some small Δx , Δy , or Δz each frame. Thus, by starting each animation and iterating over an update position command, we were able to make the object appear to be smoothly moving in the 3D environment. In our demonstration, we had 11 moveable objects: 8 bolts, the tire, the rim, and the rover itself. Each animation was mapped to voice commands through direct support from Vuforia Studio.



Figure 20: Visualizing the tire removal process

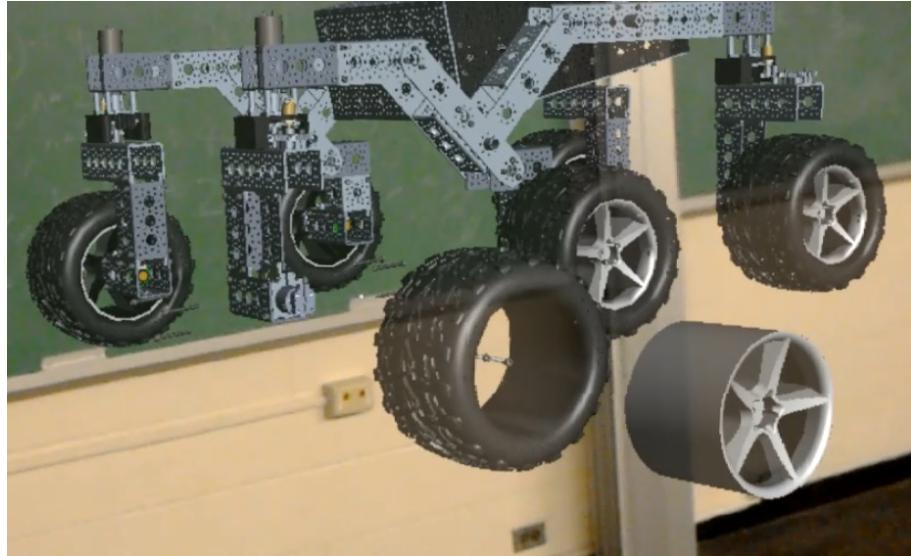


Figure 21: Visualizing the rim removal process

Again, the goal of including these visual tutorials is to provide astronauts with information not attainable through current NASA procedures and without the use of augmented reality. For example, in Figure 20, observe the internal mechanism of the wheel assembly is now visible *before* having physically taken apart the rover itself. We feel this is one of the most important advances brought about by AWARES. While EVA procedures will be practiced countless times before the EVA itself (and potentially with Virtual Reality tools instead), we feel having the ability to visualize EVA procedures during the EVA itself is paramount to a successful mission.

5.2.5 Voice Recognition

In addition to our control system attached to the spacesuit, we felt it was important to include a voice recognition aspect as well. This was primarily done for redundancy and ease of use. Given that astronauts would likely have different preferences on how to control the technology, providing multiple options to

do so will attempt to please the highest number of them. Furthermore, in the unlikely event that this aspect of our control system malfunctions, astronauts can still carry forward with the EVA and the tutorial stages. Another important reason for its implementation is the ability to map common language to more NASA specific terminology and acronyms. For example, it was shown in our final video that by using voice recognition, astronauts do not need to memorize the specific procedure to change the left front wheel of a rover, but instead can just intuitively mention something such as "Remove the left front wheel".

In terms of the implementation of voice recognition itself, Vuforia Studio provides direct support to map voice commands to animations. Thus, by mentioning certain keywords such as "Next", we make function calls to JavaScript code to begin animations.

5.2.6 Integration with NASA

Finally, it is important to consider how the augmented reality portion of AWARES would integrate into current and future NASA procedures. Our video demonstration did not show the process of placing the rover itself, so some confusion may arise as to how astronauts will actually reach the tutorial stage in the first place.

For our demonstration, we chose to manually place the rover on the ground by the use of a "Spatial Target". This is an element placed in the Vuforia Studio virtual environment that is used as a central reference for other objects. Upon loading the virtual environment to the HoloLens, users are prompted to place that spatial target somewhere in the physical environment to begin loading elements situated around it. That said, Vuforia Studio supports other placement options, most notably being the "Model Target". Model targets are used when the object is present in the physical environment. For example, given we had built the rover itself, we could scan the environment for the rover and essentially snap our hologram to the physical rover similar to what is done in this video: <https://www.youtube.com/watch?v=2ooSQmMrg4g>. This is done through rather complicated computer vision algorithms, however nicely encapsulated in one element for us through Vuforia Studio. Ideally, astronauts would have the ability to choose the type of target they would prefer prior to the EVA. For a less obtrusive experience, astronauts could place the model next to the rover itself and follow the tutorial stages this way.

In addition, a full implementation and integration would see a much more robust set of voice commands. While the tutorial currently only supports a start feature and advancing tutorial stages, astronauts would have the full capability to replay animations, navigate to specific stages in the tutorial process, and generally have a more interactable tutorial.

6 Conclusion

The application of A.W.A.R.E.S into the EVA experience is truly capable of bringing spacewalking to a new level, just in time for space exploration to be taking off in a brand new way. We believe applying augmented reality techniques that are starting to become very prevalent on Earth to spacewalking will help provide an easier transition to the more intense spacewalks that are bound to happen in the near future. While the scope of this project was limited to a single mission example, the possibilities for both the 3D augmentation and the notification systems that A.W.A.R.E.S provides are truly limitless and can be adapted to help solve the many problems of tomorrow.

7 Appendix

7.1 Estimated Time Remaining Code

```
//Booleans representing normal status level - true if functioning normally false if emergency
CO2_status;
O2_status;
batt_status;

//Data buffers for moving average
CO2_data_buff[100];
O2_data_buff[100];
batt_data_buff[100];

//struct representing user data, averaged together before this code segment
struct userData {
    O2_usage[n][2]; //rows represent percentage each data point, 2 columns are percentage remaining
    and time left
    CO2_usage[n][2];
    batt_usage[n][2];
    int weight; //number of data samples
}

while (1) {
    refreshBuffers();

    //if everything is nominal
    if (CO2_status) {
        CO2_estimate = (userData.weight*0.05)*userData.CO2_usage[currentTime] +
            (1-(userData.weight*0.05))*CO2movingAverage(CO2_data_buff); //weighting explained in
            section 2.3
    }
    //disregard past EVAs if there is an issue
    else {
        CO2_estimate = movingAverage(CO2_data_buff);
    }

    if (O2_status) {
        O2_estimate = (userData.weight*0.05)*userData.O2_usage[currentTime] +
            (1-(userData.weight*0.05))*O2movingAverage(O2_data_buff);

    }
    else {
        O2_estimate = movingAverage(O2_data_buff);
    }

    if (batt_status) {
        batt_estimate = (userData.weight*0.05)*userData.batt_usage[currentTime] +
            (1-(userData.weight*0.05))*battmovingAverage(batt_data_buff);
    }
    else {
        batt_estimate = movingAverage(batt_data_buff);
    }

    est_time = min(CO2_estimate, O2_estimate, batt_estimate);
```

```

refreshBuffers {
    //shifts all elements in buffer to the left by one
    for (i<99) {
        CO2_data_buff[i] = CO2_data_buff[i+1];
        O2_data_buff[i] = O2_data_buff[i+1];
        batt_data_buff[i] = batt_data_buff[i+1];
    }
    //updates last value with current value
    CO2_data_buff[99] = currentCO2Usage;
    O2_data_buff[99] = currentCO2Usage;
    batt_data_buff[99] = currentCO2Usage;
}

movingAverage(buff) {
    average = 0;
    weight[100]; //increasing weights for 100 values, all add up 1 and ends with 0.10
    for (i<100) {
        average+= weight[i]*buff[i]
    }
    return average / 100;
}
}

```

7.2 JavaScript Animation Code

```

// $scope, $element, $attrs, $injector, $sce, $timeout, $http, $ionicPopup, and $ionicPopover
// services are available

var modelItemCounter = 1;
var imageCounter = 1;
var timerId = -1;
var speed = 0.03;
var modelItemVar;
var imageVar;

$scope.app.playStep = function ()
{
    modelItemVar = "model-" + modelItemCounter;
    imageVar = "3DImage-" + imageCounter;
    console.log("modelItem: " + modelItemVar);
    var xpos = $scope.view.wdg[modelItemVar]['x'];
    $scope.moveAway();
}

$scope.moveAway = function ()
{
    var update_multiple = false;

    if (timerId > -1)
    {
        clearInterval(timerId);
    }

    timerId = setInterval(function () {

```

```

    if (imageCounter == 1)
    {

        $scope.view.wdg[imageVar] ['visible'] = true;

        var xpos1 = $scope.view.wdg[modelItemVar] ['x'];
        $scope.app.params.xpos = xpos1;

        $scope.$apply(function ()
        {
            $scope.view.wdg[modelItemVar] ['x'] = $scope.view.wdg[modelItemVar] ['x'] + speed;
            console.log($scope.view.wdg[modelItemVar] ['x'])
        })

        modelItemCounter++;
        modelItemVar = "model-" + modelItemCounter;

    }

}

, timingInterval);
setTimeout(function () {
    $scope.view.wdg[modelItemVar] ['visible'] = false;
    $scope.view.wdg[imageVar] ['visible'] = false;
    clearInterval(timerId);
    modelItemCounter++;
    imageCounter++;

}
, 1000);
}

```

References

- [1] Depth of field equations. URL: <http://www.dofmaster.com/equations.html>.
- [2] Hed 2200 color lcos microdisplay. URL: <https://holoeye.com/lcos-microdisplays/hed-2200-color-lcos/>.
- [3] Nasa suits. URL: <https://microgravityuniversity.jsc.nasa.gov/nasasuits.cfm?fbclid=IwAR3qpoONqWsXmgBhAjaSsSkSrPD1os0v1Et-aHuZ7EM8Uf3xAFusAWaSWj8>.
- [4] Oct 1988. URL: <https://www.cdc.gov/nchs/data/nhanes/nhanes3/cdrom/nchs/manuals/anthro.pdf>.
- [5] Bayesian network, Feb 2019. URL: https://en.wikipedia.org/wiki/Bayesian_network.
- [6] JPL open source rover, April 2019. URL: <https://github.com/nasa-jpl/open-source-rover>.
- [7] Microsoft mixed reality toolkit - unity, April 2019. URL: <https://github.com/Microsoft/MixedRealityToolkit-Unity>.
- [8] Peripheral vision, Apr 2019. URL: https://en.wikipedia.org/wiki/Peripheral_vision.
- [9] Vuforia studio augmented reality, April 2019. URL: <https://www.ptc.com/en/products/augmented-reality/vuforia-studio>.
- [10] Scope AR. Scope ar odg: Waterpump demo, Sep 2016. URL: https://www.youtube.com/watch?v=KpwYAEH_h5I.
- [11] Jon Christian. Lockheed engineers are using hololens to build nasa's new space capsule, Oct 2018. URL: <https://futurism.com/the-byte/how-nasa-is-using-augmented-reality-to-build-its-next-capsule>.
- [12] Brian Dunbar. What is a spacesuit?, May 2015. URL: <https://www.nasa.gov/audience/forstudents/5-8/features/nasa-knows/what-is-a-spacesuit-58.html>.
- [13] Tony Greicius. New 'ar' mobile app features 3-d nasa spacecraft, Mar 2018. URL: <https://www.nasa.gov/feature/jpl/new-ar-mobile-app-features-3-d-nasa-spacecraft>.
- [14] Tom Harris. How joysticks work, Jun 2002. URL: <https://electronics.howstuffworks.com/joystick1.htm>.
- [15] Mark Wade. Shuttle emu. URL: <http://www.astronautix.com/s/shuttleemu.html>.
- [16] Erin Winick. Nasa is using hololens ar headsets to build its new spacecraft faster, Oct 2018. URL: <https://www.technologyreview.com/s/612247/nasa-is-using-hololens-ar-headsets-to-build-its-new-spacecraft-faster/>.