Command and Control Systems Widgets

Maher Ismail

NASA KENNEDY SPACE CENTER

Major: Computer Engineering

Fall 2018 Session

Date: 10-11-2018

The Engineering Directorate at NASA's Kennedy Space Center (KSC) is designing a command and control system to launch future rockets. Part of this effort involves designing new visual representations for operator displays for monitoring and commanding the Space Launch System (SLS), Launch Control Systems (LCS), and future rockets. The purpose of my semester long internship as a software developer includes the design, development, integration, and verification of widgets that will be used in the firing rooms operator console displays.

Nomenclature

DSF	=	Display Services and Frameworks
KSC	=	Kennedy Space Center
SCCS	=	Spaceport Command and Control System

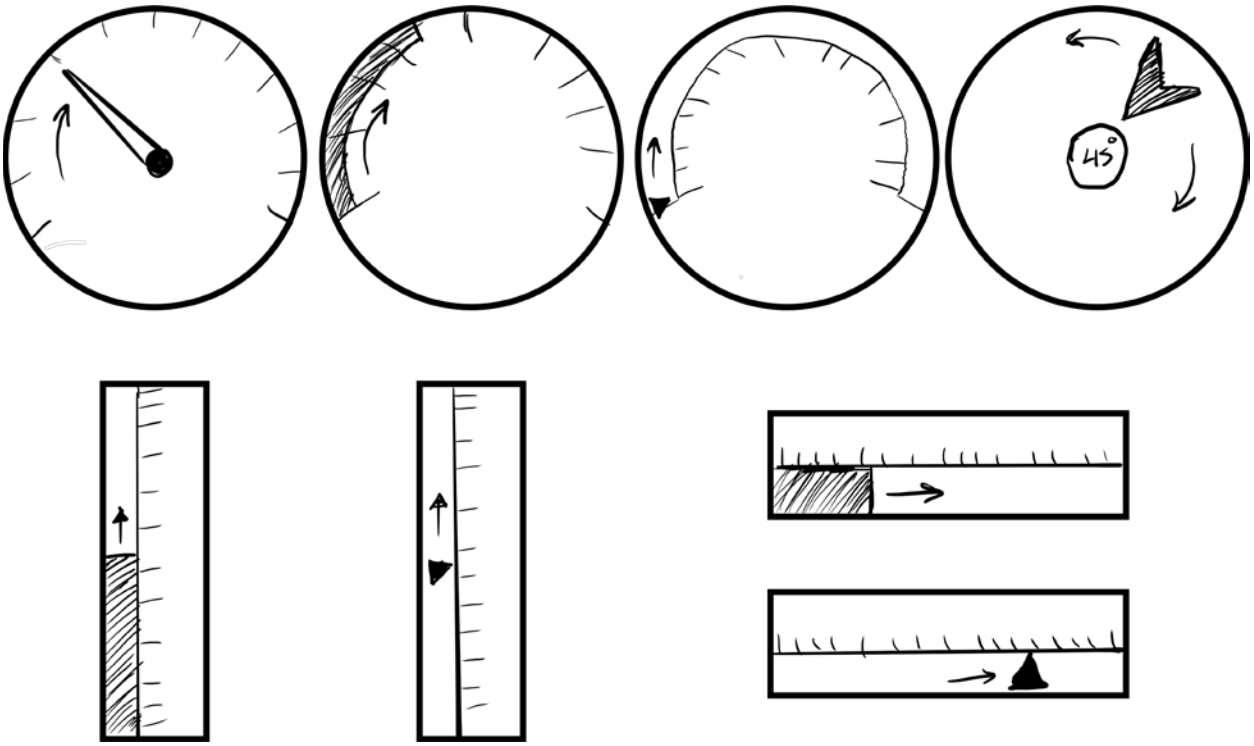
I. Introduction

The SCCS engineers at NASA KSC currently use a numeric representation to show different qualities and attributes of the vehicle and ground data on launch day. The engineers using these displays have requested a more visual representation of the data. The type of visual representation used for this effort is widgets. The widgets would allow the engineers to take in significantly more information at once with much more efficiency and ease. Instead of just showing the value of a specific element and its unit, a gauge could be used to show where that value lies in a specified range, like a speedometer in a car. The engineers have requested several widgets including bar fill, radial fill, bar pointer, radial pointer, and wind direction. The widgets were also envisioned to change colors depending on the severity state of the values they represent. Having these widgets will provide an enhanced situational awareness on console as to how close a measurement is to a limit and can potentially make launch day operations run more smoothly.

II. Objectives

Various gauges can be created to visually represent vehicle and ground system data. The gauge choice for each measurement will be determined by the type of data the gauge will represent. It is up to the designer of the display to select the desired gauge based on both personal preference and type of data to be displayed. The following gauges can be used interchangeably with the option to use more than one gauge to represent a given measurement: a radial pointer gauge that rotates from left to right, a radial fill gauge that fills up from left to right, a needle pointer that points at the value inwards with a small indicator while

rotating from left to right, a bar fill that fills upwards, a pointer fill that moves upwards, a bar fill that fills from left to right, a pointer fill that moves from left to right, a circular wind direction widget that points inwards with the value in the center displaying angle values in the range from 0 to 360.



Gauge Sketch by Maher Ismail

After talking to the engineers, they requested a set of specific features that can optionally be applied to any of the widgets. Not all features are applicable to all the widgets so they were designed each having their own set of features. Some of these features include: allowing the object's range to adapt the maximum and minimum values that can later be changed, adjusting severity signs dynamically (colored indicators to where warning starts), adding a pulsing severity color affect, allowing more parts of the widgets to pulse during red severity, inserting a label on the widget and allow it to be dynamically changed, adjusting the number of tick marks (main and intermediate) dynamically, allowing minimum and maximum values to be changed dynamically, using default colorblind colors for all colors shown, allowing the user to turn off severity signs, having the option to toggle the flashing of severity colors, adding the ability to invert all colors, adding an option for an invalid state, allowing widget to be resizable, allowing widgets to take colors directly from the system, and dynamically adjusting the font of numbers on scale.

III. Approach

A. Learning the Software

Since there were no area experts for the tool I was working on still working the project, the only way for me to learn how to use the software was reverse engineering existing widgets while exploring more options and looking online for articles and tutorials. After a few days, I learned that the software was similar in many ways to software I have used in the past, and that allowed me to learn it much faster.

B. Design

The design of the widgets was requested to be minimalistic, not having too many distracting colors and not being too graphically advanced, so as to not distract the user when there are many of the widgets on the screen at once. The design also needed to be easy on the eyes so the user would not experience unnecessary eyestrain. Using these instructions, I made a series of prototype designs and showed them to mentors, interns, and engineers. I then polled them for input on pros and cons of each design. I compiled their input into a combined list of all the positive aspects of each and modified the design, constantly improving it with feedback from the engineers who made the request for the widgets. As time passed, the engineers kept requesting functional and aesthetic improvements until they were satisfied with the design.

C. Functionality

To make the pulsing effect, I let the desired color blink between different colors with a high frequency. I added twenty one colors and then decremented the RBG value by thirty until I reached five, then incremented by thirty to create the pulsing effect. To make the needle and the pointer move clockwise, I bound the object from a minimum value of -20 to a maximum value of 200, then oriented it to the center of the widget. Since when an angle is increased, it moves counter clockwise from zero to 180, any value must be multiplied by -1 to move in the opposite, clockwise direction. To allow the object to change depending on the parameter, the following formula was inserted in the rotation value on the object

$$(Value - InputMinimum) * \frac{200 - -20}{InputMaximum - InputMinimum} + -20$$

For the radial fill, the only way to produce that same effect was to create an arc and allow the size to change depending on the incoming value. The arc fits perfectly in the gauge but it only has two attributes, a start angle and a delta angle (how far it goes). Since the arc was not designed to move dynamically, it only increased in size counterclockwise so the size was always added to the left side of the starting angle. This was incorrect, since the gauge was intended to fill clockwise, like any other mechanical gauge. When I tried to add a negative delta angle, the arc always became a full circle. Therefore, any negative number was not accepted and there was no way to change the direction of the fill. Instead of changing direction, I allowed the size to increase counter clockwise while simultaneously moving the starting angle the same amount so as to create the illusion of filling up clockwise. It also needed to be bound by a specific visual range and an incremental range depending on what the user chose as maximum and minimum values. This resulted in the following formulas to calculate the start angle and the delta angle:

Start Angle: $(200 - Value) * \frac{200 - -20}{InputMaximum - InputMinimum}$

Delta Angle: $(Value) * \frac{200 - -20}{InputMaximum - InputMinimum} + 1$

For the bar fills I added the same formulas above but inserted it in the size value instead of rotation value. The formula changes the height of the fill for the vertical bar and the width of the fill for the horizontal bar was added. For the pointer fills, the same formulas were used, but instead of size they changed the position value: Y position for the vertical pointer and X position for the horizontal pointer fill. Later I noticed an issue with the needles or bars exceeding the widget's enclosure if their runtime values exceeded the chosen range.

To fix this issue, when the value was higher than the maximum, I replaced every occurrence of maximum values in the formulas with the value itself. If the value was lower than the minimum, I replaced every occurrence of the minimum value in the formulas with the value itself. This also solved the issue of the arc changing into a complete circle when given a number lower than zero.

D. Perfecting the widget

As I became more familiar with the software, it was easier to understand its capabilities. After creating the prototype, I was able to implement additional features and modify aesthetic properties to meet user requirements. I was able to get a better understanding of what the user needs and how it will work in real time, thus providing more relevant widgets. Since these widgets were designed with constant input from the users, the engineers may be able to reuse these widgets on future projects.

E. Connecting widgets to the System

To connect the newly designed widgets to the existing displays, I attempted to try to "trick" the system into thinking I was using one of its already existing widgets. I gave it the same name as an existing widget and observed the result. Although this technique didn't work, it gave me insight into how the widgets take information from the other programs, allowing me to dissect the widget to see which parts were essential for it to function. Knowing which part is essential allowed me to know where to look and what I needed for my widgets to function in a similar way. After a lot of troubleshooting, and with the help of some of the engineers who work closely with this program, I was able to connect the widgets to the system as well as control it dynamically with different parameters.

IV. Conclusion

The use of widgets rather than numerical data can increase productivity and attention during launch activities. The widgets allow the engineers to spot abnormalities easier, thus getting a jump on fixing problems. Once testing is complete, theses widgets will be used for the launch of Exploration Mission 1 .This internship allowed me to apply my computer, problem solving, and creative skills. Combining and further developing these skills towards something that will be used to further the space program has been a great privilege. Being able to contribute to this mission is one of the highlights of my technical career.

Acknowledgements

I would like to thank my mentors Jill Giles and Jamie Szafran. I would also like to thank Jonathan Serrano, Mario Blalock, and Thong Tran for always supporting me and answering any questions. Additionally, I would like to thank Jason Kapusta for always providing guidance, as well as Jordan Kiser. Furthermore, Oscar Brooks for his career development input and branch supervision. Finally, I would like to thank Gwendolyn Gamble and Kathleen Wilcox from the NASA KSC Education office, for being friendly and patient with all the interns and supporting us throughout the whole process.