Indian Journal of Science and Technology, Vol 8(16), DOI: 10.17485/ijst/2015/v8i16/64243, July 2015
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Generation of Symbols on Primary Flight Display and Its Validation under Host Environment

N. Amali Aarthy* and R. Ezhilrasie

Department of Computer Science and Engineering, SASTRA University, Thanjavur - 613401, Tamil Nadu, India;
amalirithi14@gmail.com, ezhil@cse.sastra.edu

Abstract

The aim of this paper is to develop the symbols (graphical view of flight indicators) of Primary Flight Display page at the host environment and to validate the symbols using GUI (Graphical User Interface) at run time. The symbols of these indicators can be developed using OpenGL libraries which gives no compatibility problems at the target system (Helicopter). With the support of these libraries, the symbols for the indicators can be developed. In the host environment, the data required for the symbols of flight indicators can be given manually by the developer at the compile time or at the run time using GUI. Based on the given data, symbols can be displayed and the developer can check the correctness of the symbols. The process of giving data (periodic data) to the program at the compile time and at the run time can be compared with each other. The method of initialising the structures at the compile time increases the number of times the program being executed. So the GUI is developed with all the required controls which can receive all the valid and invalid constant values at run time. It reduces the number of executions of the program. Thus improves the validation efficiency. With the support of developed GUI, the program can be executed for one time the symbols on OpenGL window can be validated against all the valid and invalid values. Thus improves the efficiency of validating the correctness of symbols by the developer. This process can be applied in the development and validation of other pages of IADS (Integrated Architecture and Display System).

Keywords: Compile Time, IADS, Indicators, PFD, Run Time, Symbols,Validation

1. Introduction

In this paper, the generation of symbols on PFD[1] page are explained and methods to validate the working of symbols are explained. The symbols[2,3] of PFD page can be developed in the host environment. Once the symbols of PFD page are built, before it gets loaded in the target system, it can be checked for its correctness by passing valid inputs and invalid inputs (valid inputs and invalid imitates the real time values) either at compile time or at run time. If the values are passed at the compile time then for each set of inputs, the program needs to be executed. This increases the number of times the program being executed. It degrades the efficiency of validation.

The valid and invalid inputs can be passed to the program at the runtime with the support of GUI[4] (Graphical User Interface) which allows the developer to test[5,6] the symbol with many sets valid and invalid inputs at a time. This reduces time consumption in validating the symbols. Thus it improves the efficiency of validation.

2. Host Environment

The host environment is the computer system which is used to build the software needed for the helicopter. It can be provided with operating system and tools which can be used to build the software for the helicopter. In this paper, the host environment is provided with Windows 7 OS, Visual studio 2012 express for desktop, and OpenGL libraries.

*Author for correspondence

3. Proposed System

The flight key indicators[7] (Glass Cockpit[8]) are mainly classified as circular, tape indicators. The main fight key indicators on the PFD page are as follows:

- Air Speed Indicator (displays the helicopter's air speed in km/h or k (knots),
- Power Bucket (gives enough information about the engines),
- Vertical Velocity Scale (displays the helicopter vertical speed in meters/min rate of dive/climb),
- Barometric Altitude Indicator (displays the helicopter barometric altitude in meters/feet units),
- Radar Altimeter Display (display the altitude as received from the helicopter's RAD ALT),
- Hover Performance Display (display the rotor performance),
- G meter (displays the normal acceleration).

4. Circular Indicator

The airspeed indicator is an example for circular indicator which can be represented using a circle symbol (Figure 1) with short and long plain line measurement.

4.1 Role

This indicator is used to display the airspeed of the helicopter in Km/h or K.

Figure 1. Airspeed Indicator.

4.2 Periodic Data

The periodic data are the real time data received from the sensors which are processed by the Digital and Mission computer[9] (DMC) which is the heart of IADS[9] and it will be sent to relevant indicator. In host environment, it is impossible to work on the real time data. So a structure is created which comprises all the periodic data related to each indicator. The structure can take the following form:

Code 1: Sample periodic data structure for airspeed indicator

```
typedefstruct
{
float as;
floatVneON;
floatVneOFF;

}
asDatatype;
```

The values for this defined structure can be assigned either at the compile time or at the run time. The values can be passed from the windows' controls like text box which is placed on the form (GUI) to the relevant periodic data at the run time. This can be done as follows:

Code 2: Defining text box's function

```
public:System::VoidtextBox1_TextChanged(System::
Object^sender, System::EventArgs^  e)
{
int temp;
temp=System::Int32::Parse(textBox1->Text);
asdatatype.as=temp;
}
```

Then the defined periodic data structure takes the following form:

Code 3: Assigning values to the periodic data structure from values entered in text box

```
asDatatype d1=
{
asDatatype.as=temp,
asDatatype.VneON=temp1,
asDatatype.VneOFF=temp2
};
```

Figure 2. Symbol of Airspeed Indicator with GUI.

The Figure 2, shows the openGL window with the created form which acts as a GUI. The values that are entered in the controls of GUI will be reflected on the openGL window.

5. Tape Indicator

The vertical velocity scale indicator can be represented as a tape – like symbol with a long vertical thick plain line with alternative short and long plain line measurements at the left or right side of the symbol. These symbols can be generated using openGL functions based on the end user requirements.

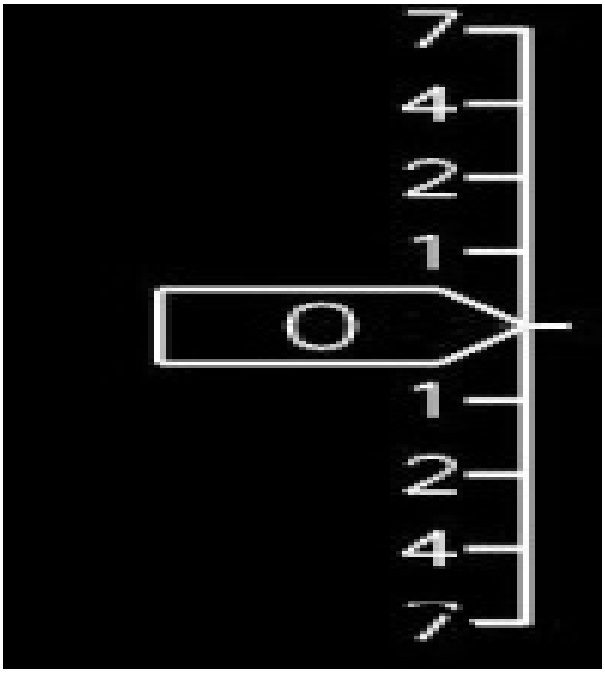

Figure 3. Vertical Velocity Scale.

5.1 Role

This indicator shows the vertical speed of the helicopter meters/min rate of dive/climb.

The Figure 3, shows the tape indicator which consist of vertical scale with plain lines with numbers beside the lines and boxed arrow mark with digital readout.

5.2 Periodic Data

The structure is created which comprises all the periodic data related to vertical velocity scale indicator. The structure can take the following form:

Code 4: Sample periodic data structure for vertical velocity indicator

```
typedefstruct
{  floatvvelocity;

}
asDatatype;
```

Code 5: Defining text box's function

```
public:System::VoidtextBox1_TextChanged(System::
Object^sender, System::EventArgs^  e)
{
int temp;
temp=System::Int32::Parse(textBox1->Text);
asdatatype.vvelocity=temp;
}
```

Then the defined periodic data structure takes the following form:

Code 6: Assigning values to the periodic data structure from values entered in text box

```
asDatatype d1=
{
asDatatype.vvelocity=temp,
};
```

The Figure 4 shows the OpenGL window which displays the symbols and the GUI. The data entered in the controls of the window form (GUI) can be passed to the OpenGL window and the symbols will be displayed based on the received data.

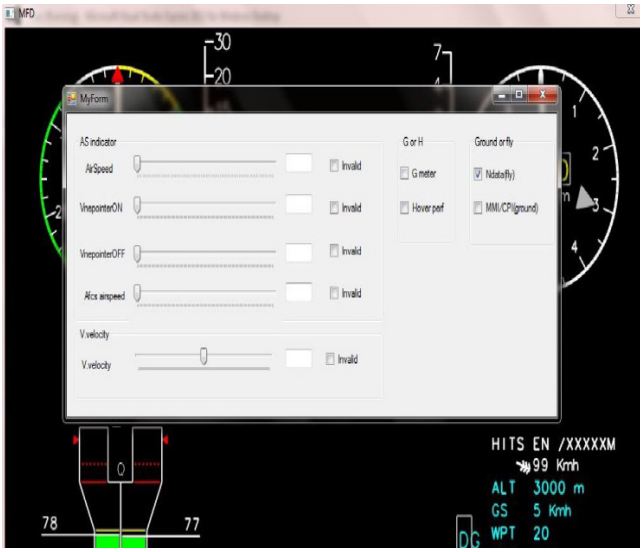

Figure 4. GUI for vertical velocity scale indicator along with other indicators.

6. Performance Analysis and Results

The process of assigning values to the periodic data structure at the compile time and at the run time (with the help of GUI) is compared with each other. This comparison is based on the number of times the program needs to be executed to validate[10] the symbols at the run time. The efficiency is improved, when values are assigned at the run time. Thus it decreases the number of executions of the program.

From the Figure 5, it is shown that assigning values at the run time reduces the number of executions which will improve the validation efficiency.

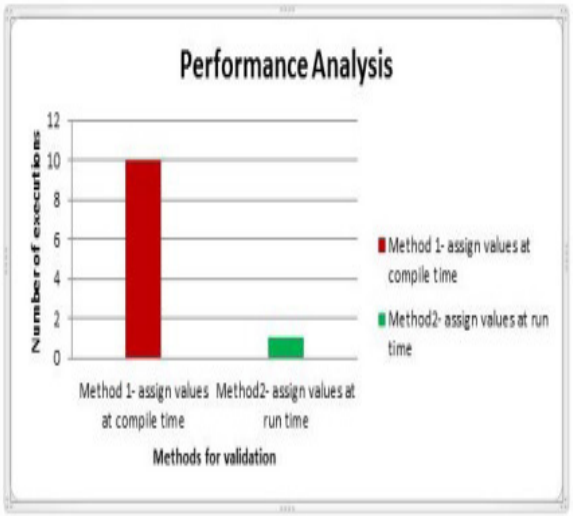

Figure 5. Comparison of validation methods.

7. Conclusion

The values assigned to the periodic data structure at the run time improve the validation efficiency. Thus it reduces the time consumption for validation. This method can be applicable for reading values of the parameters related to indicators such as radius, font colour etc. at the run time. This allows the end user to modify the indicators using GUI without altering the developed code.

8. References

1. Campbell CB, editor. Advanced integrated general aviation primary flight display user interface design, development and assessment. 2002 Proceedings of the 21st Digital Avionics Systems Conference; IEEE; 2002.
2. Bartolone AP, Hughes MF, Wong DT, Takallu MA, editors. Symbology development for general aviation synthetic vision primary flight displays for the approach and missed-approach modes of flight. Proceedings of the Human Factors and Ergonomics Society Annual Meeting; SAGE Publications; 2004.
3. He G, Feyereisen T, Wyatt S, editors. FMS flight plans in synthetic vision primary flight displays. SPIE Defense, Security, and Sensing; International Society for Optics and Photonics; 2009.
4. Arunachalam A. Innovative approach of testing in Event Driven Software (EDS). Indian Journal of Science and Technology. 2014; 7(S5):37–40.
5. Jacob TP, Ravi T. An optimal technique for reducing the effort of Regression Test. Indian Journal of Science and Technology. 2013; 6(8):5065–9.
6. Gharibi A, Khaki R. A Method for Calculating Aircraft Aerodynamic Loads Using Obtained Data from Flight Simulation. Indian Journal of Science and Technology. 2015; 8:74.
7. Helicopter Components, Sections, and Systems. Available from: www.faa.gov/regulations_policies/handbooks_manuals/aviation/helicopter_flying_handbook/media/hfh_ch04.pdf
8. Introduction of Glass Cockpit Avionics into Light Aircraft. Available from: www.ntsb.gov/safety/safety-studies/Documents/SS1001.pdf

9. Available from: www.asian-defence.net/2010/10/hal-dhruv-weapons-system-integration.html
10. Nanda M, Jayanthi J, Rao S. An effective verification and validation strategy for safety-critical embedded systems. International Journal of Software Engineering and Applications (IJSEA). 2013; 4(2).