Scientific Visualization, 2021, volume 13, number 1, pages 124 - 137, DOI: 10.26583/sv.13.1.09

Safety critical visualization of the flight instruments and the environment for pilot cockpit

B.Kh. Barladian[1,A], A.G. Voloboy[2,A], L.Z. Shapiro[3,A], N.B. Deryabin[4,A], I.V. Valiev[5,A], S.V. Andreev[6,A], Yu.A. Solodelov[7,B], V.A. Galaktionov[8,A]

[A] Keldysh Institute of Applied Mathematics RAS
[B] State Res. Institute of Aviation Systems (GosNIIAS)

[1] ORCID: 0000-0002-2391-2067, bbarladian@gmail.com
[2] ORCID: 0000-0003-1252-8294, voloboy@gin.keldysh.ru
[3] ORCID: 0000-0002-6350-851X, pls@gin.keldysh.ru
[4] ORCID: 0000-0003-1248-6047, dek@keldysh.ru
[5] ORCID: 0000-0003-2937-8480, piv@gin.kldysh.ru
[6] ORCID: 0000-0001-8029-1124, esa@keldysh.ru
[7] ORCID: 0000-0001-5891-7645, yasolodelov@2100.gosniias.ru
[8] ORCID: 0000-0001-6460-7539, vlgal@gin.keldysh.ru

Abstract

The article is devoted to the pilot display visualization system for cockpit of a civil aircraft. Different content of modern pilot displays is discussed. The peculiarity of visualization system development for avionics is considered. All software used in civil aviation systems is safety critical and must comply with international safety standards. This imposes additional requirements both on the hardware used and the software development process. The core of the pilot display visualization system is OpenGL Safety Critical (SC) library. In the paper both the software and hardware OpenGL SC implementations elaborated by us are presented. We describe the aspects of rendering speedup by optimizing of OpenGL SC codes for the specifics of aviation application, by usage of processor multiple cores and, finally, by elaboration of library exploiting the GPU hardware acceleration. Achieved rendering speed measured for real aviation applications is reported in the paper. Only relatively simple applications can be rendered at an acceptable frame rate without using a GPU. Also further development and possibility of the visualization system certification are discussed. The elaborated visualization software is intended for use with the Russian real-time operating system JetOS.

Keywords: pilot display, glass cockpit, flight instruments visualization, OpenGL Safety Critical.

1. Introduction

In recent decades civil aircraft cockpits become significantly more complex. Aircraft equipment and systems have improved operational performance and some aspects of situational awareness. At the same time they significantly raised the requirements to pilot qualifications. An increase in the capacity of passenger ships, an increase in the duration of flights in adverse weather conditions also increase the psychological burden on the pilots. The significant pilot mental efforts are required to keep track of the number of indicators and instruments. This necessitated the development of a new display concept. Accordingly, both the indicators and the layout of the display panel should be user-friendly to improve the interaction between the pilot and the aircraft [1, 2]. Poor interaction between the pilot and the interface in airplanes can lead to accidents. In this regard, the issues of design and implementation of instruments

in the cockpit are of great importance from the point of view of ensuring the safe and efficient work of pilots.
Traditionally, one display or dial was dedicated for one flight instrument. But with introduction of Electronic Flight Instrument Systems a modern aircraft have got “glass cockpit”. This new interface ideology allows to improve flight data perception [3] by combining important information into one multi-functional display which provides integrated, easily comprehensible picture of aircraft. Nowadays, information from several sources is visualized on one large screen at once (for example, the dashboard of the MC-21 aircraft shown in Fig. 1). Moreover the display can be configurable and contain different information in different flight segments. The study [4] discovered that while most of novice pilots have strong subjective preference for the glass cockpit they demonstrated poorer performance on test flights using it compared to the traditional cockpit displays. This emphasizes importance both the glass cockpit design and high performance of display system.

Fig. 1. Dashboard of the MC-21 aircraft.

It is necessary to note an important aspect of the information visualization in the cockpit. Most of the widgets used to represent various information in the modern aircraft cockpit are standardized. There are several reasons for this. Firstly these widgets have already been tested and made comfortable by a significant part of the pilots. Secondly it makes it easier for pilots to retrain from one type of aircraft to another one (this issue was investigated in [5]). Thirdly most cockpit interfaces are designed using ANSYS SCADE Display tools [6], which offer a number of pre-built widgets for displaying typical information on cockpit displays.
Modern aircraft on-board systems are based on the concept of Integrated Modular Avionics (IMA) [7, 8]. The concept idea is integration of hardware devices and embedded processors in one network operated by a real-time operating system (RTOS). This concept together with usage of modern equipment (for example, a LED display instead of mechanical dials) leads to reducing number of devices and cables, reducing the aircraft weight that results in more effective fuel consumption. Therefore implementation of the visualization component for aircraft has its own additional specifics related to the equipment used on board and operating system.
One more very important aspect of the design of onboard cockpit systems (including pilot display rendering) is that they belong to the class of systems critical from the safety point of view. A critical safety or life-saving system is a system whose failure or malfunction could result in one of the following: death or serious injury to people, loss or serious damage to equipment / property or damage to the environment. When designing such systems for an aircraft cockpit, it is necessary to comply with the requirements of such standards as ARINC 653 [9] for the operating system and OpenGL SC (Safety Critical) [10] for visualization system.

2. Aircraft flight instruments visualization

A large amount of flight information must be visualized on the cockpit displays. This information, coming from a wide variety of devices, must be displayed at the same time. The primary flight display (PFD) is used in modern aircraft to display most important flight information. FAA regulation describes that a PFD must include an airspeed indicator, turn coordinator, attitude indicator, heading indicator, altimeter, and vertical speed indicator. The details of the display layout on a primary flight display can vary enormously, depending on the aircraft, the aircraft's manufacturer, the specific model of PFD, certain settings chosen by the pilot and various internal options specified by the aircraft's owner. However the great majority of PFDs follows a similar layout. Examples of layout on a primary flight display developed for Sukhoi Superjet and MC-21 are presented in the Fig. 2 and 3.

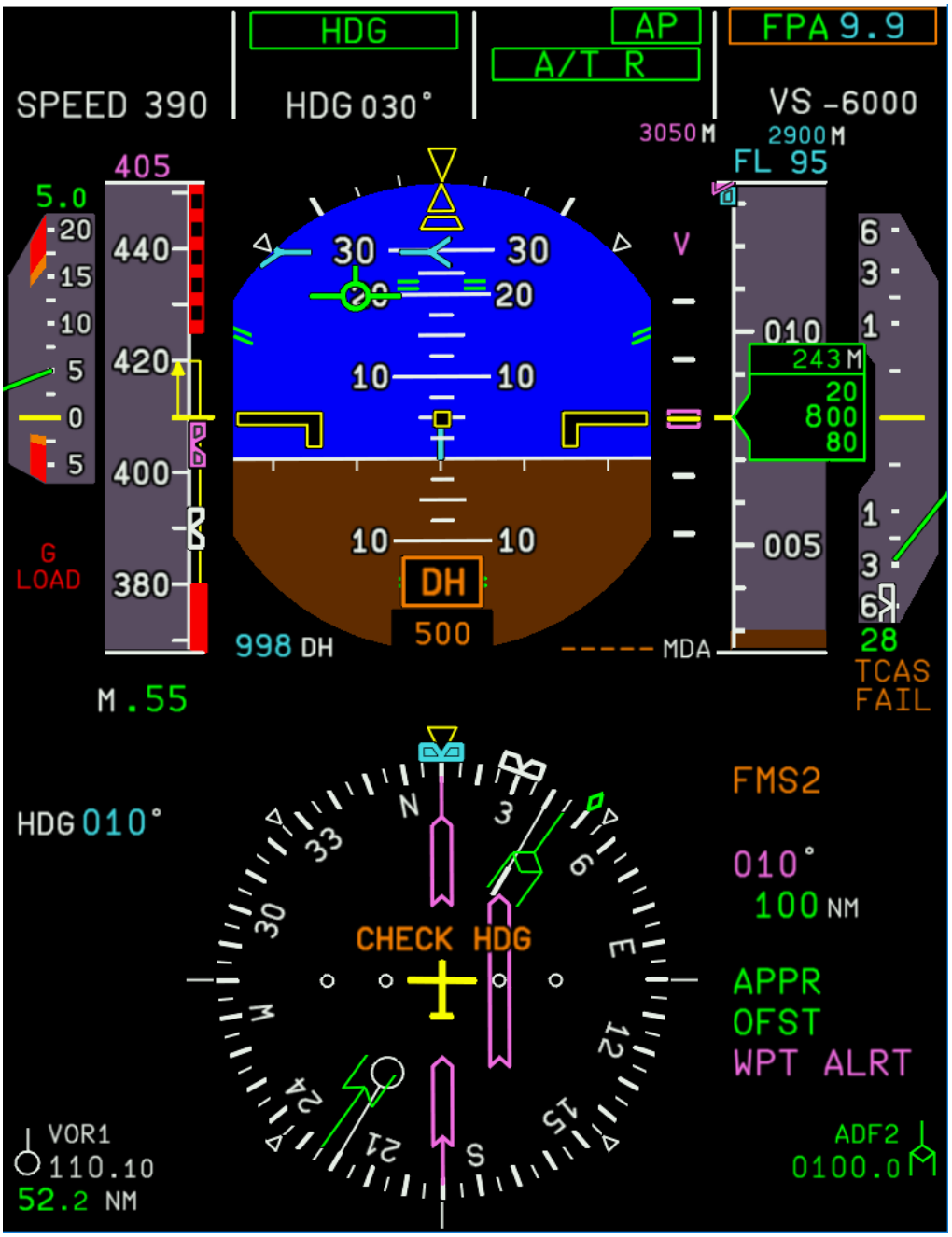

Fig. 2. PFD developed for Sukhoi Superjet aircraft.

Fig. 3. PFD developed for MC-21 aircraft.

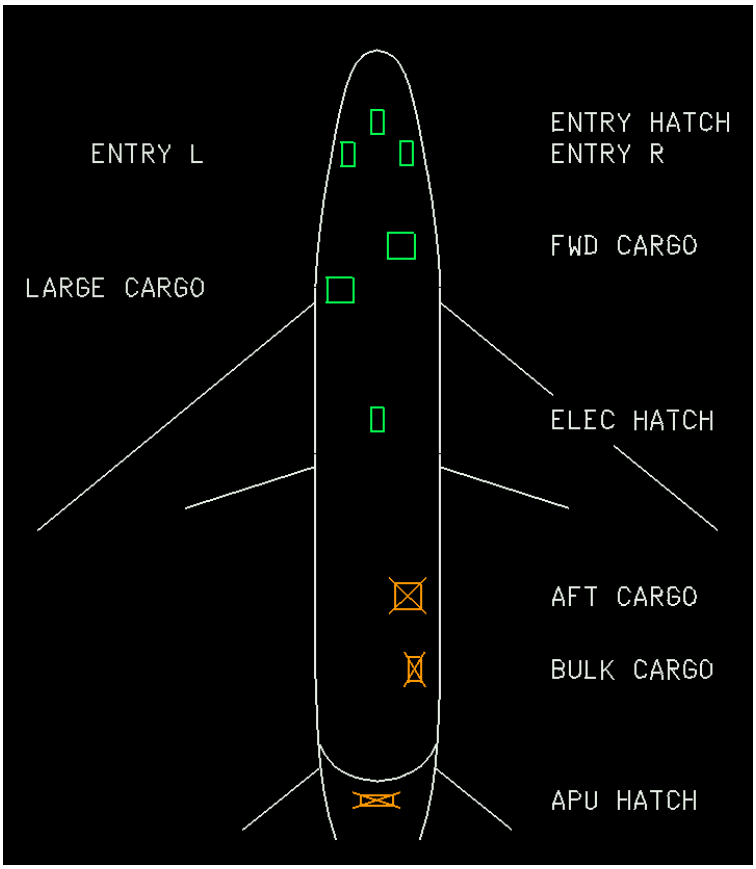

Fig. 4. State of the aircraft doors.

Additionally the number of widgets can be used to display such technical characteristics as engine speed, oil pressure, fuel quantity, various pneumatic, hydraulic and electrical circuits. Also there are various kinds of warnings and alarm signals which should be displayed on screen. An example of the widget showing the “open” or “closed” state of the aircraft doors is shown in Fig. 4.

3. Environment visualization

In addition to the flight instruments the pilots also need information about the environment, such as a navigation map, a weather map, other aircrafts nearby, relief of the surrounding area, as well as detailed information about the aerodrome where they plan to land. Providing this information is a perspective way of civil aircraft glass cockpit improvement. More and more aircrafts are supplied by wide LCD displays where the environment can be visualized.
An important tool that is always present on cockpit displays is the Navigation Display (examples are shown in Fig. 5 and 6). The Navigation Display (ND) screen shows the aircraft's lateral navigation status. Typically it has the ability to display different modes and functions: route flown, VHF omni-directional radio range, Instrument Landing System, Weather Radar (like it is shown in Fig. 5), TCAS (Traffic alert and Collision Avoidance System) etc.

Fig. 5. Airbus navigation display prefixed waypoints

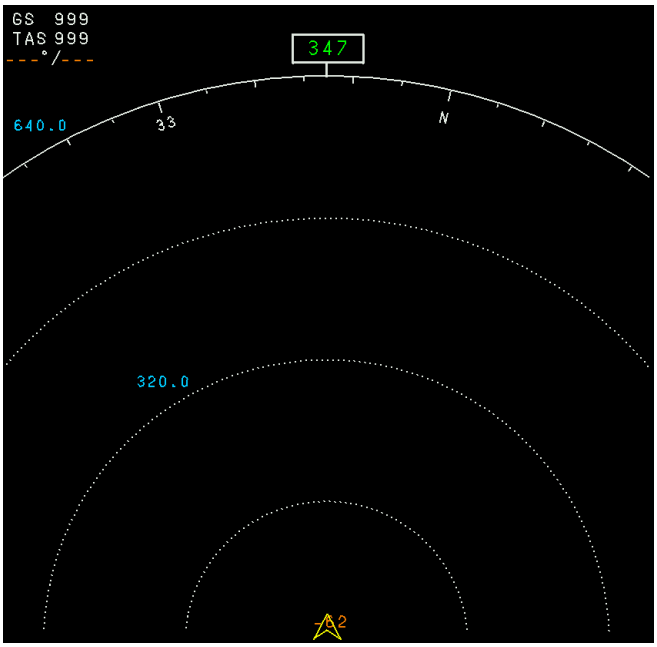

Fig. 6. Navigation display (ND).

Pilot's request for such visualization is growing while its implementation is rather complex problem. The situation results in some uncertified additional tools which pilots use on small private jets. For example, the tool from Horizon company [11] was elaborated by pilots and can be installed on tablets. Of course this product is not elaborated under avionic standards and is not safety critical. So it can be used only as additional device in private flights and under responsibility of the pilot. But it demonstrates what pilot considers as useful and helpful visualization. Fig. 7 presents 3D Synthetic Vision which provides an indication of the aircraft's attitude as well as the area surrounding the aircraft such as terrain, nearby ChartElements, Runways and so on. The terrain is visualized in warning colors depending on vertical distance to it. Useful flight information is displayed in a PFD like style (such as groundspeed, altitude, vertical speed, destination etc.).

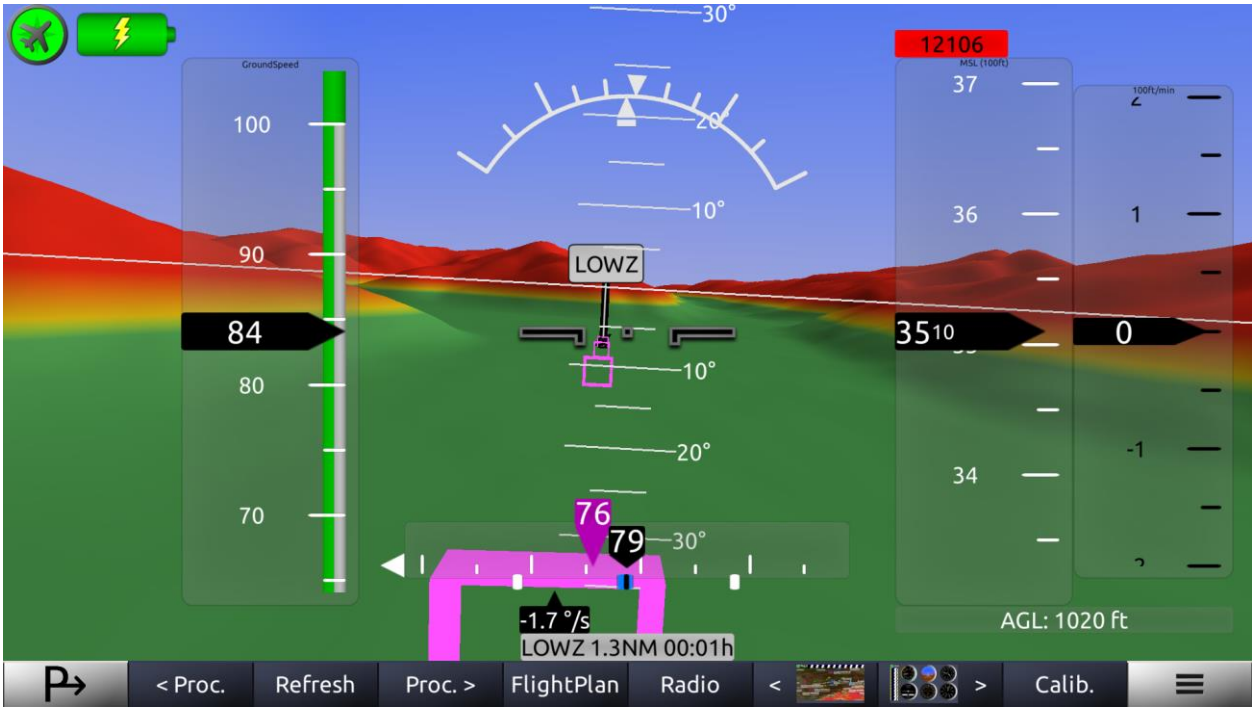

Fig. 7. 3D Synthetic Vision from the Horison company.

The company also proposes other screens containing environment information in visual form. For example, moving map can display landmarks, streets and highways, rivers and lakes, country borders, airport and taxiway charts, etc.

4. Peculiarity of the elaboration of the aircraft visualization system

The implementation of any onboard software should take into account the specifics of the hardware. As a rule, aircraft onboard systems do not use the latest high-speed processors but reliable processors that have been used for many years and for which the problems of "childhood diseases" have already been solved. Typical examples are QorIQ PowerPC processors [12] with Embedded Radeon™ E4690, E8860 GPUs [13] or NXP i.MX 6 SoC Processor Family [14] with Vivante Graphics Processors [15]. These processors have a relatively low energy consuming and performance. Therefore to obtain the required rendering speed the software implementation of visualization system must be optimized for the class of applications used in the aircraft cockpit.

Most interfaces of civil aircraft cockpit are currently being developed using the ANSYS SCADE Display tools [6]. The code generated by this system assumes the use of the OpenGL library for visualization of interface widgets. Since the visualization system must be Safety Critical, then the OpenGL SC library should be used. Currently, there are two significantly dif-

ferent OpenGL SC standards in use: OpenGL SC 1.0.1 and OpenGL SC 2.0. The significant difference between OpenGL SC 1.0.1 and OpenGL SC 2.0 is presence of programmable shaders in 2.0. Some functionality version 1.0.1 is removed in 2.0 and can be replaced with shader programs. In other words, OpenGL SC 2.0 emphasizes a programmable 3D graphics pipeline versus the fixed functionality of OpenGL SC 1.0.1. The high level differences between the two graphics pipeline architectures are shown in Fig. 8 [16]. One can see that it is significant change going from OpenGL SC 1.0.1 to OpenGL SC 2.0. With OpenGL SC 2.0 you need to manage vertex transformations, lighting, texturing using OpenGL Shading Language (GLSL) vertex and fragment shaders.

Fig. 8. Differences between fixed graphics pipeline and programmable graphics pipeline

Thus, to develop the visualization system for the cockpit we need to implement the OpenGL SC library according to one of these standards. To ensure certification of the developed equipment, the entire design process must comply with the standard DO-178C [17], Software Considerations in Airborne Systems and Equipment Certification is the primary document by which the certification authorities such as FAA, EASA and Transport Canada approve all commercial software-based aerospace systems. Access to the source code is a main requirement for the software certification process. This is why no proprietary binary libraries can be used in the development of onboard equipment. In particular, this applies to the OpenGL library needed to visualize the state of the aircraft and the environment on cockpit displays.
For Safety Critical onboard cockpit systems we have to use RTOS which satisfies ARINC 653 standard. ARINC 653 does not permit using dynamic memory allocation and releasing, also it does not permit multithreading. Fortunately we developed our visualization under JetOS [18], which supports the special extension of ARINC 653 that allows the use of multicore processors. This extension is called Asymmetric Multi-Processing (AMP). So we could improve the rendering speed by using multicore processors without deviating from the standards. There is an important difference between the AMP technology and multithreading used un-

der Linux to accelerate visualization on multicore computers [19]. While multithreading ensures the efficient concurrent work of several threads in the same address space the AMP technology in JetOS supports running several modules (i.e., JetOS instances) on different processor cores independently of other modules. This approach guarantees the safety and reliability required for the onboard software. However it is less efficient than the multithreading technology. In addition the use of AMP technology requires significantly different organization of parallel computing.

5. Proposed solutions

As it was mentioned above the main part of pilot display visualization system is OpenGL SC library. There are two ways to implement it for avionic needs. The first way is its purely software implementation and the second one is to use the GPU hardware support. Certainly the software library will be slower than hardware one. However the software solution is much easier to optimize for special applications [20]. Also the software solution is simpler to certify. We have implemented both versions of the OpenGL SC library.

A lot of algorithms for the software implementation of OpenGL are described in the literature, for example, [20-22]. Its implementation on modern powerful computers, like Intel i7-4770 3.4 GHz, gives quite acceptable rendering speed for typical aviation applications: ~ 50 frames per second. The speed is close to the values shown by the OpenGL driver for the NVIDIA Quadro 410 video card. Unfortunately the same implementation of the OpenGL library on a typical aircraft computer based on the PowerPC e500mc processor, 4 cores, 1 GHz [12] or NXP i.MX6 [14] does not provide the required performance. For example, the speed for PFD application shown in Fig. 3 is less than 2 fps on these computers.

Such rendering speed is unacceptable. Thus we had to develop own implementation of OpenGL SC and to optimize it for avionics applications to ensure an acceptable rendering speed. Our optimization was based on the applications used for visualization in the cockpit displays. Most of them use the visualization of 2D objects only. Mainly they visualize the readings of various devices in digital and analog forms, the spatial attitude of the aircraft, various indicators, meteorological and cartographic data. Such visualization uses a limited number of combinations of OpenGL instructions. Basing on this fact we managed to considerably speed up rendering for dedicated applications.

In avionics applications the greater part of the visualization time is taken by rasterization algorithms. We focused on the rasterization procedure acceleration and succeeded by developing the following approaches [19]:

1. To compute various quantities (Z-coordinates, color, or texture coordinates) inside a triangle we used linear interpolation instead of directly barycentric coordinates calculation. This accelerated the rasterization approximately by a factor of 1.9.
2. Whenever possible we used fixed point arithmetic instead of floating point one. This accelerated the rasterization approximately by a factor of 2.2.
3. For value computations in a row of pixels inside a triangle we used a set of special intrinsic functions. The operations to be done for a given pixel depend on a number of conditions which take a significant amount of time to check. But these conditions are the same for all pixels of the triangle. So we can check this condition once and then apply the specified special function for all pixels inside triangle without additional checking. This accelerated the rasterization approximately by a factor of 1.7.

These approaches provide up to 7x faster rendering which is still insufficient for most practical aviation applications. Since both processors (that are supposed to be used in aircraft) have multiple cores the parallelizing of visualization procedure is a natural way to speed up rendering. In [22] the multithreading under Linux was used for this purpose but as was said above multithreading is not permitted for avionics. So we elaborated own parallel computing technology for parallelization of OpenGL SC using allowed AMP technology. The main idea of our approach is that the application runs on one processor core and the rendering of sequential

frames occurs in parallel on other processor cores. The synchronization developed by us ensures the correct rendering of the prepared images on the display. The details of our solution are described in [23]. Parallel visualization using AMP technology gives additional rendering acceleration from 1.5 to 3.4 times depending on application and processor. The ultimate limitation of speed is caused by copying an image from the processor memory to the screen procedure which cannot be done in parallel.

Parallelization using AMP technology gives acceleration but nevertheless achieved rendering speed still is not satisfactory for some typical applications. Moreover for most of the analyzed applications satisfactory speeds were achieved only when using four processor cores. But this is not always permissible since there are other consumers of computing resources in the onboard system. For these reasons we elaborated hardware based OpenGL SC implementation which uses the hardware acceleration of promising platform i.MX6 with the Vivante GPU. Implementetion of the hardware OpenGL SC library was based on the open source MESA package [24]. The MESA package cannot be used directly under JetOS RTOS. Its adaptation requires significant efforts to solve the following problems:

1. Only appropriate JetOS special functions can be used to allocate memory.
2. Memory allocation is allowed only at the partition initialization stage.
3. Dynamic memory allocation, releasing memory is prohibited. This means that the corresponding MESA calls must be excluded or rewritten using own memory manager.
4. Any multithreading in accordance with the ARINC 653 specification cannot be used. Therefore the MESA calls using mutex objects should be excluded or rewritten.
5. All code redundant for OpenGL SC must be excluded according to DO-178C regulations.

After the MESA adaptation we provided support for the OpenGL SC versions 1.0.1 and 2.0. It required development of a code generator which creates code in accordance to the header file of corresponding library version. The detailed description of problems of the hardware OpenGL SC implementation and our solutions to them are in [25]. It should be also noted that certification of OpenGL SC library with hardware acceleration is more difficult task then for software one.

6. Results and discussions

Both developed OpenGL SC versions, software and hardware ones, were tested on the same set of aviation applications. Besides Sukhoi Superjet and MC-21 PFD (Fig. 2 and 3) we used several test applications: GlassCockpit (Fig. 9), relatively simple Counters and FlightDisplay, terrain visualization (Fig. 10), Doors (shown on Fig. 4), Navigation display (ND, Fig. 6) and the Aerodrom Moving Map (AMM) shown on Fig. 11. We compared rendering speeds on the i.MX6 platform using the software implementation of the OpenGL SC 1.0.1 (SWGL) and the hardware implementation of OpenGL (HWGL). Test results are shown in Table 1. Software OpenGL was implemented with usage of 1 (SWGL 1 column) or 4 (SWGL 4 column) processor cores.

Table 1. Rendering speed (in frames per second) of applications for various OpenGL implementations.

	SWGL 1	SWGL 4	HWGL
SSJ_PFD (fig. 2)	5.9	13.8	10.8
MC21_PFD (fig. 3)	6.3	15.6	20.0
Counters	23.9	35.4	60
GlassCockpit (fig. 9)	10.5	28.7	29.7
FlightDisplay	9.7	26.4	60
Terrain (fig. 10)	7.7	20.9	60
Doors (fig. 4)	12.0	34.7	60
ND (fig. 6)	22.5	40.3	60
AMM (fig. 11)	5.6	9.8	15.3

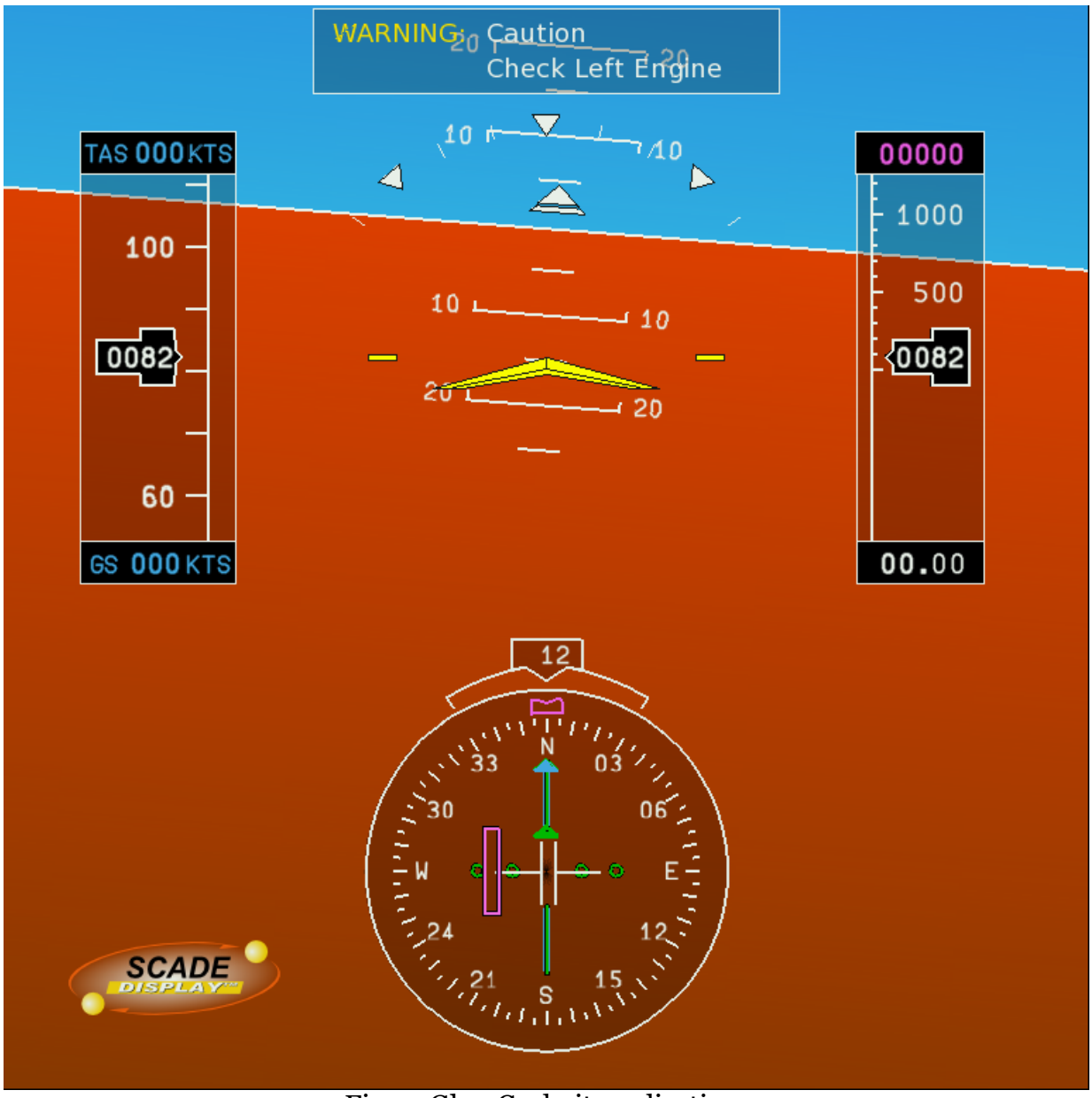

Fig. 9. GlassCockpit application.

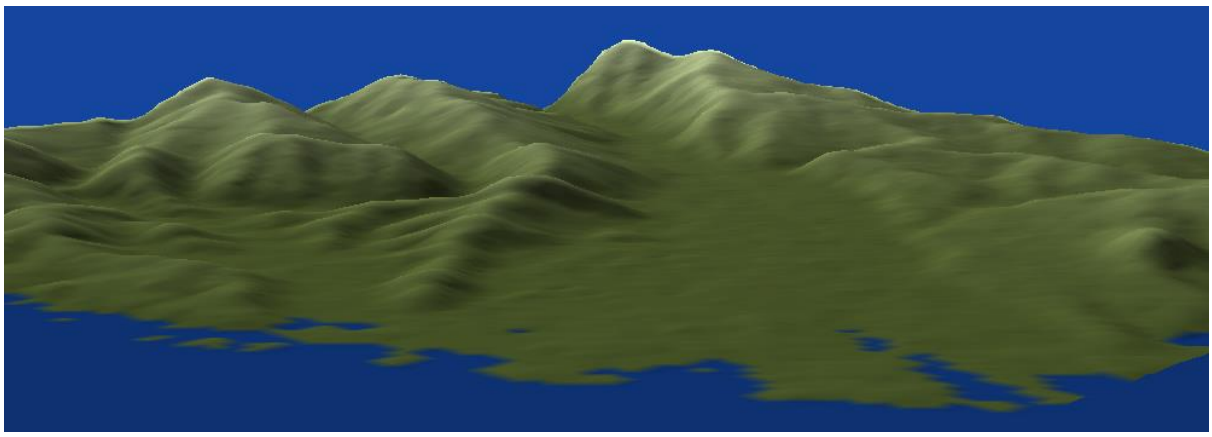
Fig. 10. Terrain visualization application.

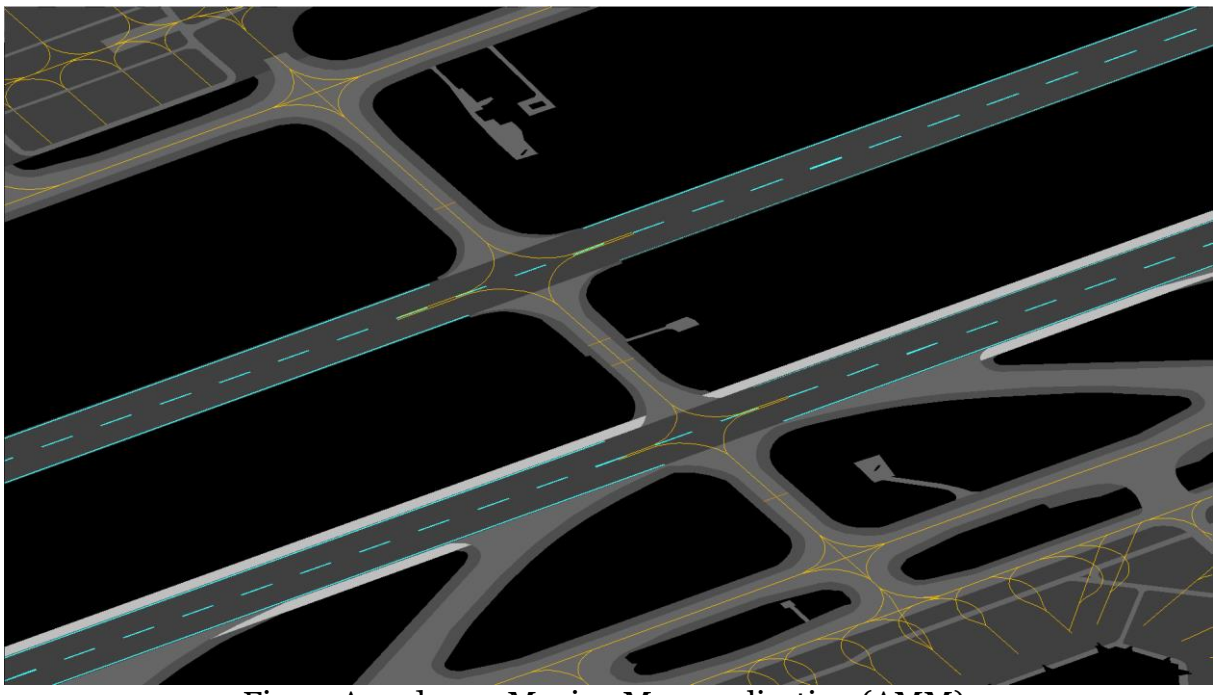

Fig.11. Aerodrome Moving Map application (AMM).

It should be noted that the 60 frames per second shown in the table for some applications is limited to synchronization with the display refresh rate. The real speed of hardware OpenGL SC library for these applications is higher. Thus the use of the Vivante GPU support for the OpenGL SC 1.0.1 library implementation under the JetOS operating system can achieve significant rendering acceleration.
It should also be noted that the obtained rendering speed for the SSJ PFD and AMM applications is insufficient for practical use in onboard systems. Analysis of the AMM application showed that it calls the hardware OpenGL in inefficient way. It used 1227 calls to the glDrawElements() function to render one frame. When we united groups of primitives of the same color it turned out that it was enough to use only 7 calls of the glDrawElements() function to draw one frame. Rendering speed increased from 15.3 to 60 frames per second. Hardware accelerated OpenGL has too much overhead for calling the glDrawElements() function: loading data into the GPU. Similar problems are for the SSJ PFD application. Rendering a single frame uses more than 600 calls to the glDrawArrays() function. The application code was generated using the SCADE package. In general it can be optimized but this not a simple task.

7. Conclusion

Two versions of the OpenGL SC library – software and hardware ones – were implemented by us. They work under the JetOS real-time operating system which complies with theARINC 653 standard. Certainly performance of the software library is lower than hardware one. However the software solution can be much easier optimized for specific applications and its certification is much easier. The studies showed that the rendering speed of software OpenGL SC library is not satisfactory for some application even if parallel processing via AMP technology is used.
The hardware OpenGL SC library was implemented for prospective processor i.MX6 with the Vivante GPU. In most cases the hardware acceleration of the Vivante processor allowed to get acceptable rendering speeds for onboard equipment. However in some cases the effective uses of OpenGL SC with hardware acceleration requires optimization or even rewriting of the application code taking into account the specifics of the GPU pipeline. Also the question about possibility to certify the hardware OpenGL SC code against DO-178C standard remains open yet. In reality this is challenging task. But its solution will allow us to create high performance visualization system for the pilot displays in aircraft cockpit.

References

1. Şenol, M.B.: A new optimization model for design of traditional cockpit interfaces, Aircraft Engineering and Aerospace Technology, 2020, **92**(3), 404-417. https://doi.org/10.1108/AEAT-04-2019-0068
2. Thomas, P., Biswas, P., Langdon, P.: State-of-the-Art and Future Concepts for Interaction in Aircraft Cockpits, Lecture Notes in Computer Science, 2015, **9176**, 538-549. DOI: 10.1007/978-3-319-20681-3_51
3. Kal'avsky, P., Rozenberg, R., Mikula, B., Zgodavova, Z.: Pilots' Performance in Changing from Analogue to Glass Cockpits, In Proc. of the 22nd Int. Scientific Conf. on Transport Means (Transport Means), 2018, pp. 1104-1109.
4. Wright, S., O'Hare, D.: Can a glass cockpit display help (or hinder) performance of novices in simulated flight training? Applied Ergonomics, 2015, **47**(1), 292-299. https://doi.org/10.1016/j.apergo.2014.10.017
5. Socha, V., Socha, L., Hanakova, L., Valenta, V., Kusmirek S., Lalis, A.: Pilots' Performance and Workload Assessment: Transition from Analogue to Glass-Cockpit, Applied Science, 2020, **10**(15), 5211. https://doi.org/10.3390/app10155211
6. ANSYS SCADE Display https://www.ansys.com/products/embedded-software/ansys-scade-display. Last accessed 05 Mar 2021
7. Fedosov, E.A.: A project of creating new generation integrated open architecture modular avionics, Polyet, 2008, no. 8, pp. 15–22.
8. Fedosov, E.A., Kos'yanchuk, V.V., Sel'vesyuk, N.I.: Integrated modular avionics, Radioelektron. techn., 2015, no. 1, pp. 66–71.
9. ARINC Standards Store. https://www.aviation-ia.com/product-categories. Last accessed 05 Mar 2021
10. Khronos OpenGL SC Registry. https://www.khronos.org/registry/OpenGL/index_sc.php Last accessed 05 Mar 2021
11. Horizon. Your portable glass cockpit. https://helios-avionics.com. Last accessed 05 Mar 2021
12. QorIQ® Processors Based on Field Proven Power Architecture Technology P-Series. https://www.nxp.com/products/processors-and-microcontrollers/power-architecture-processors/qoriq-platforms/p-series:QORIQ-POWER-ARCHITECTURE-P-SERIES Last accessed 05 Mar 2021
13. CORE AVI Supported Graphics Processors. https://coreavi.com/wp-content/uploads/CoreAVI-ProductBrief-ArgusCore-SC1.pdf Last accessed 05 Mar 2021
14. i.MX 6 Series Applications Processors. https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/i-mx-applications-processors/i-mx-6-processors:IMX6X_SERIES Last accessed 05 Mar 2021
15. Vivante Corporation. https://en.wikipedia.org/wiki/Vivante_Corporation Last accessed 05 Mar 2021
16. White Paper: Should You Be Using OpenGL SC 2.0?, https://coreavi.com/wp-content/uploads/2018/08/OpenGL2_White-Paper_2016.pdf Last accessed 05 Mar 2021
17. DO-178C. https://en.wikipedia.org/wiki/DO-178C Last accessed 05 Mar 2021
18. Mallachiev, K.M., Pakulin, N.V., and Khoroshilov, A.V., Design and architecture of real-time operating system, Trudy ISP RAN, 2016, **28**(2), 181-192.
19. Barladian, B.Kh., Voloboy, A.G., Galaktionov, V.A., Knyaz', V.V., Koverninskii, I.V., Solodelov, Yu.A., Frolov, V.A., Shapiro, L.Z.: Efficient Implementation of OpenGL SC for Avionics Embedded Systems, Programming and Computer Software, 2018, **44**(4), 207-212. DOI: 10.1134/S0361768818040059
20. Dudra, J., Mileff, P.: Advanced 2D rasterization on mordern CPUs, Applied Information Science, Engineering and Technology: Selected Topics from the Field of Production Information, 2014, 7(5).

21. Wihlidal, G.: Optimizing the Graphics Pipeline with Compute, In Proc. of Game Developers Conf., 2016.
22. Laine, S., Karras, T.: High-performance software rasterization on GPUs, In Proc. of High-Performance Graphics, Vancouver, 2011, pp. 79-88.
23. Barladian, B.Kh., Shapiro, L. Z., Mallachiev, K.A., Khoroshilov, A.V., Solodelov, Yu.A., Voloboy, A.G., Galaktionov, V.A., Koverninskii, I.V.: Visualization Component for the Aircraft Real-Time Operating System JetOS, Programming and Computer Software, 2020, **46**(3), 167-175. DOI: 10.1134/S0361768820030020
24. The Mesa 3D Graphics Library. http://www.mesa3d.org Last accessed 05 Mar 2021
25. Barladian, B.Kh., Deryabin N.B., Voloboy A.G., Galaktionov V.A., Shapiro L.Z.: High speed visualization in the JetOS aviation operating system using hardware acceleration, In Proc. of the 30th Int. Conf. GraphiCon'2020, St. Petersburg, Russia, CEUR Workshop Proceedings, 2020, V. 2744, pp. short3:1-9, DOI: 10.51130/graphicon-2020-2-4-3