Development Environment of 3D Graphics Systems

● Yuya Tagami ● Makoto Watanabe ● Yuko Yamaguchi

Recently, more and more in-vehicle devices have been using 3D graphics in applications such as car navigation systems and digital meters. These devices adopt user interfaces that are more graphical, and they have come to offer visual effects that were previously impossible to produce. Fujitsu Semiconductor is focusing on augmented reality, in which actual images taken by cameras are combined with 3D graphics for integration with the real world to enhance the vision of users, and is moving ahead with the development of graphics systems. It requires considerable expertise to build and develop these systems and applications that use 3D graphics. Fujitsu Semiconductor is developing and offering a development environment that can reduce the burden on application developers using 3D graphics and software to realize products capable of delivering optimum functionality and performance. This paper describes the software configuration and architecture needed for a system that uses 3D graphics and gives examples of their application.

1. Introduction

Recently, the phrase "three-dimensional (3D) graphics" has been attracting attention in many fields. With touch panels used in devices such as tablet PCs, representations with smooth movement of a beautiful graphical user interface (GUI) on an LCD have been realized. Applications that offer visual effects and intuitive finger operation of objects are very user-friendly and various applications that use them are becoming popular. Such representations have been made possible by 3D graphics technology.

In addition, systems are being researched and promoted that enhance the vision of users by making use of representations with 3D graphics. In the field of medical imaging, for example, video images from MRI, CT, ultrasonic devices, and such like are represented with 3D graphics to allow medical staff to observe patients from unrestricted viewpoints.

For embedded systems, OpenGL ES has been established by the Khronos Group as a standard application programming interface (API) for drawing 3D graphics. The Khronos Group has established various other industry-standard APIs as well.[1)] Fujitsu Semiconductor offers graphics libraries compliant with OpenGL ES 2.0.

This paper first describes the system configuration for 3D graphics representations. It goes on to explain the software architecture, basic principle of the implementation technology, application examples and issues. Lastly, future prospects are described.

2. Graphics system

For the development of applications and systems that make use of 3D graphics, a number of types of expertise such as real-time system and image processing are required. To represent augmented reality (AR) with 3D graphics, for example, real-time images acquired by multiple camera video inputs are combined with 3D models. In this process, AR systems check the overall balance of the input images and carry out image correction processing, such as correcting a region where visibility is reduced by applying a light source and accentuating points to note. Subsequently, the video subjected to image processing is combined with 3D models and the viewpoint from which it is to be seen is specified for drawing. Finally, the completed video

is displayed on the screen [**Figure 1 (a)**]. This entire processing must be completed within the period of video capture from the cameras to minimize the delay, thereby building it as a real-time system.

Controlling and constructing all of these processes from applications requires substantial labor and is difficult. For this reason, software that allows users to easily develop applications without the need for expertise is desired. In addition, applications that use 3D graphics require advanced elaboration. Therefore, there are expectations for a simulator on a PC that enables users to confirm operations equal to the embedded system before real development. Under these circumstances, Fujitsu Semiconductor offers a development environment that reduces the burden on users in application development and allows easy development of applications using 3D graphics [**Figure 1 (b)**].

A variety of technologies must be linked to build a graphics system with user applications. MB86R11[2], which is a graphics SoC for in-vehicle applications, has multiple graphics engines such as 3D graphics, 2D graphics and image processing engines for realizing various graphics applications.

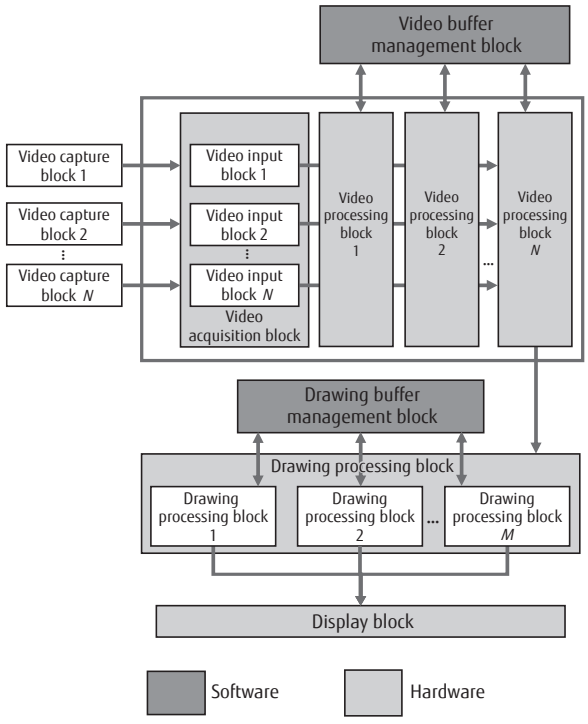
In the system, these graphics engines are controlled independently of the CPU and can be run in parallel. The states of the individual graphics engines are managed by their own drivers and, even when they are used from multiple tasks, the user can run them without being aware of other tasks.

3. Combination of 2D and 3D graphics

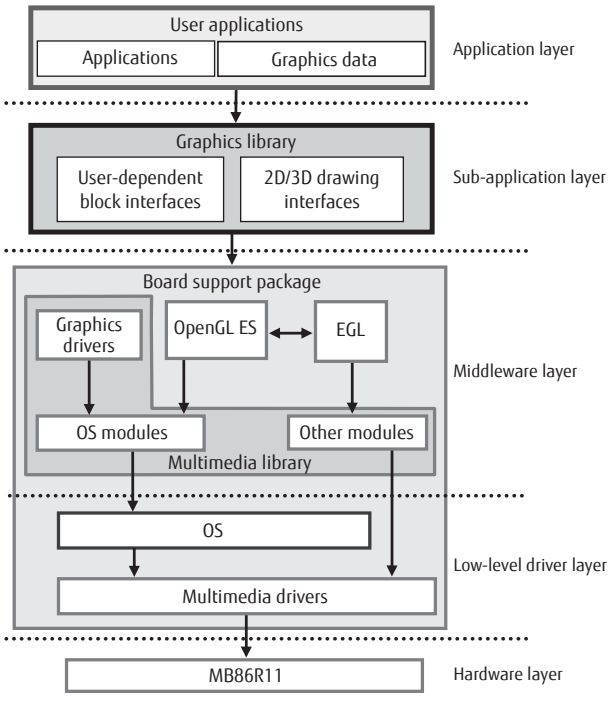
This section presents the graphics processing technologies implemented by Fujitsu Semiconductor.

To bring out sufficient system performance that can handle the increasing application complexity and resolution, systems must be able to control all graphics engines so that they run efficiently. For that purpose, we have provided software such as graphics libraries and graphics drivers, which are offered together with graphics SoCs, with various contrivances.

For the graphics library for MB86R11, OpenGL ES 2.0 has been adopted as one graphics interface. Fujitsu Semiconductor's graphics SoCs support OpenGL ES 2.0, with fixed pipelines eliminated, to keep up



(a) Processing flow



(b) Graphics software configuration

Figure 1
Graphics system.

with the evolution of shading algorithms and have the 3D graphics engine equipped with a programmable shader. Programmable shaders are capable of pixel-by-pixel processing and can realize unlimited representations without any special graphics engine.

Programmable shaders, which can also be used for image analysis and numerical computation as general-purpose computing on graphics processing units (GPGPU), are expected to serve as a technology that improves application performance.

3D graphics engines can represent images that are similar to those produced by 2D graphics engines, not to mention advanced representation. However, 2D data created in advance with a PC may be processed at a higher resolution and higher speed. For that reason, 2D graphics are drawn at a high speed by providing a separate 2D graphics engine so as to improve the system performance.

Generally, 2D/3D graphics engines can only accept and execute independent drawing processing commands. Accordingly, in a system configuration to control multiple graphics engines, control for parallel processing of graphics engines by the CPU was complicated. This restricts the processing speed when multiple applications are run.

Meanwhile, MB86R11 integrates command sequencers that queue drawing processing commands in order to reduce the load of controlling graphics engines. The command sequencers can accept more than one of the series of commands that control the graphics engines and are capable of running the graphics engines without interruption.

With this mechanism, the command sequencers can take over some of the interrupt processing that was conventionally performed by the CPU, which reduces the load on the CPU. For efficient operation of these command sequencers, the software provides control by building commands that can be run while detecting when the graphics engines have finished drawing.

MB86R11 integrates the same number of command sequencers as the graphics engines. The software has processing commands constructed for the respective command sequencers so that the parallel operation between the graphics engines is not hampered. When 2D and 3D graphics are mixed in one drawing buffer, the command sequencers can be controlled with commands to wait for each other to finish drawing, without running them in parallel (**Figure 2**).

In this way, the results of drawing by the 2D/3D graphics engines are ultimately output to the display by means of the display controller. Generally, representation of multiple images superimposed on each other can be achieved by superimposition in the memory by means of a graphics engine and superimposition in the

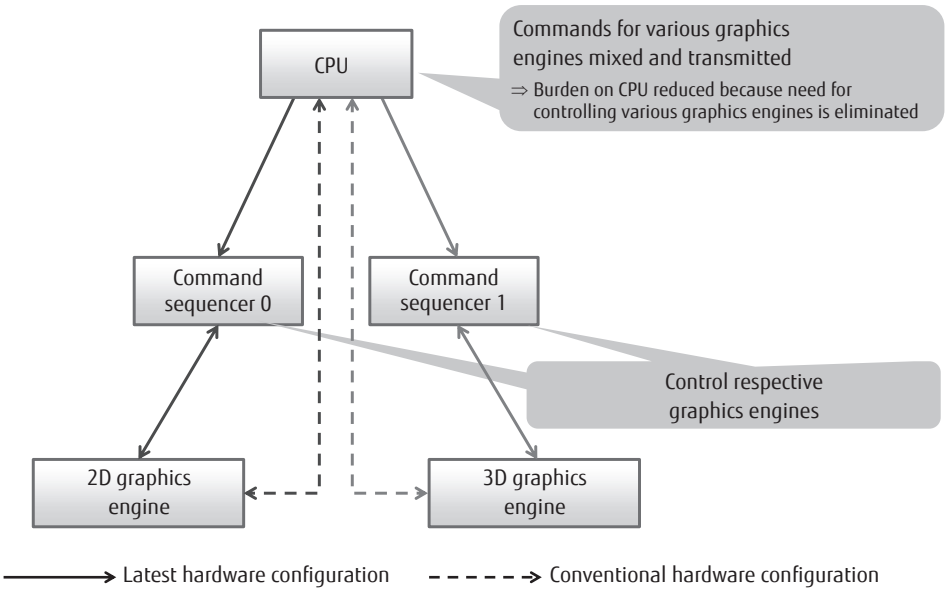


Figure 2
Command sequencer operation.

display controller.

Fujitsu Semiconductor's graphics SoCs have already been provided with a function to draw multiple images on multiple layers and superimpose the layers by using the display controller. Superimposition by means of a display controller is effective especially when images with different refresh frequencies are superimposed (**Figure 3**).

At a certain point of time of an application, for example, there may be both layers requiring and not requiring refreshing. In this situation, software can be used for control to prevent unnecessary refreshing.

If the display controller is overloaded, it is effective to have images superimposed in the memory. Which of the graphics modules is loaded depends on the application. For that reason, the software is capable of designing applications according to the system load to prevent disproportionate loading on any specific graphics module.

In this way, Fujitsu Semiconductor provides graphics software with contrivances that achieve the optimum balance between linking with various devices, drawing, and display refreshing with a limited amount of resources.

4. Combination of camera video images and graphics

For a system that includes camera video input, the result of processing must be output without delay at the specified frame rate regardless of the load of image processing that takes place internally.

In graphics drawing processing, the load varies for each frame. Accordingly, simply synchronizing the system operation with the timing of graphics drawing processing causes uneven sampling intervals of the camera input video. To avoid this phenomenon and use camera video sampled at regular intervals as the input video, the capture controller is provided with a triple buffer configuration including a reference and double buffers.

Data delivery between graphics modules uses double or triple buffering. Even when the output result of a graphics module is being referenced by another graphics module, drawing or writing in parallel is

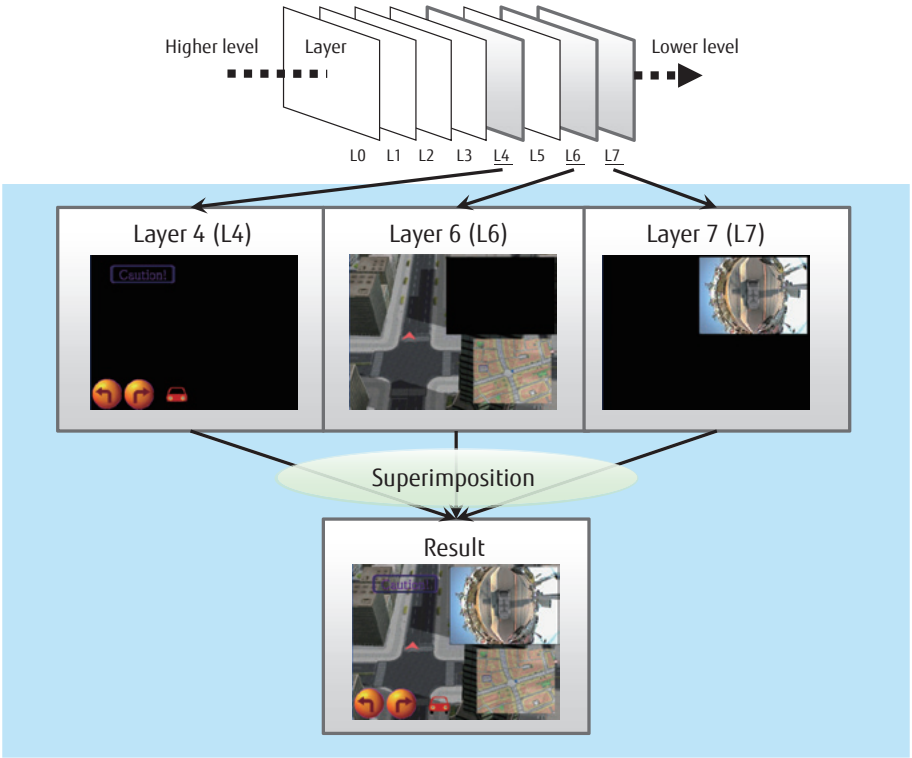


Figure 3
Layer superimposition by display controller.

allowed (**Figure 4**).

This mechanism enables an application to acquire a camera input image at an arbitrary timing. It means that, for example, running the application in synchronization with a certain clock can generate output images using camera input video at regular intervals.

In such a system in which camera input images are handled by an application, multiple pieces of linked software process the input images from the capture block through graphics drawing to the display module shown in Figure 4. Regarding the software, integrated designing from the application layer to the driver layer with the specifications and characteristics of the hardware taken into consideration prevents buffer copies, which may cause performance degradation. The structure is optimized for the individual graphics modules to constantly run in parallel.

5. Application example

Fujitsu Semiconductor has commercialized on MB86R11 a wraparound 3D monitor system[3),4)] (**Figure 5**) researched and announced by Fujitsu Laboratories. The wraparound video imaging technology has utilized the image processing technologies described earlier and been built to provide an integrated system.[5),6)]

The wraparound 3D monitor is a video processing technology for visibility enhancement to be used in vehicles, and it three-dimensionally combines video images taken around the vehicle and shows on the display the results of observation from unrestricted viewpoints. To switch between displayed video images, the viewpoint can be moved smoothly (Figure 5).

The wraparound 3D monitor system is required to draw and display numerous polygons in real time and, for efficient processing, we used vertex buffer objects (VBOs) to reduce the amount of memory transfer. Meanwhile, the programmable shader is used for camera video processing that requires changing for each frame. The speed and drawing quality is maintained in this way.

The wraparound 3D monitor system composes images by joining multiple camera video images but large brightness differences may be generated between cameras depending on the direction of sunlight, causing unnatural images. To address this problem, the brightness levels of the individual cameras are averaged and the joints are smoothed by using the programmable shader to offer video images with improved visibility.

6. Future issues

The field of 3D graphics has a lot more potential including, for example, representations that are intuitively easy for the user to understand and new representations with a factor of entertainment. With the wraparound 3D monitor system, research has been in process on a technology that allows users to view blind spots hidden from their own vehicle by capturing not

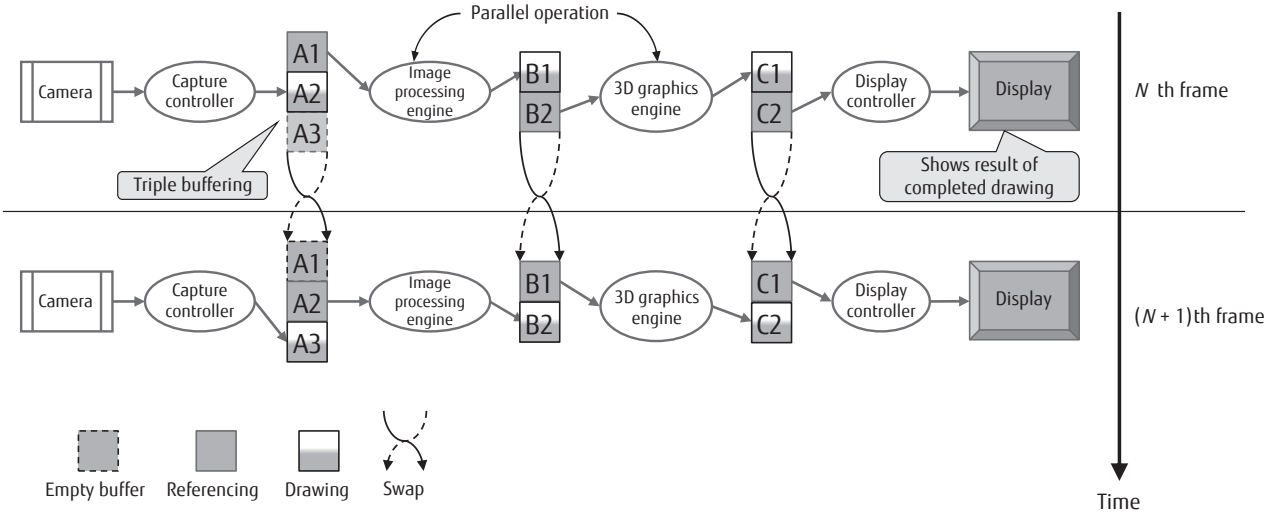


Figure 4
Image of parallel operation of graphics modules.

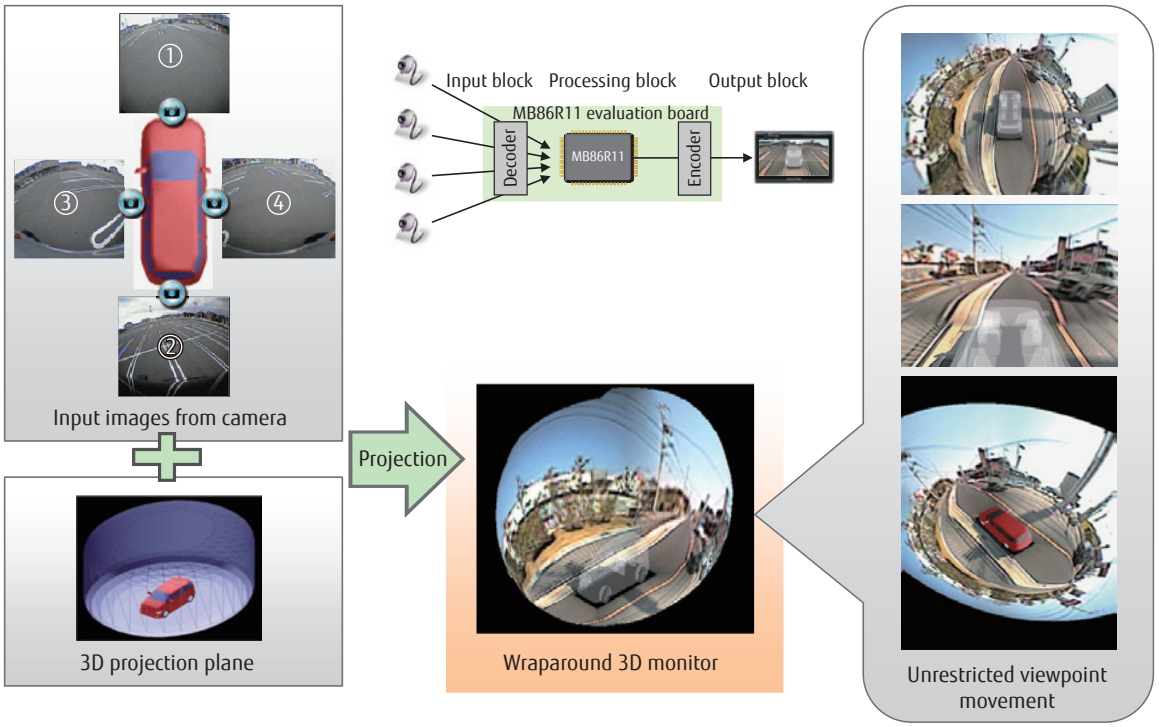


Figure 5
Wraparound 3D monitor system.

only video images from the cameras of their own vehicle but also those from other vehicles or surveillance cameras and combining them. The video images captured with the cameras are projections of the 3D real world onto a 2D plane. By restoring the actual distance perspective and stereoscopic effect from these video images in real time, 3D video images that are closer to reality can be created, which is another technology under research.

Representation of AR or virtual reality (VR) by using 3D graphics as described above essentially requires speed performance and image quality that the user finds comfortable. The issues to be resolved to that end include:

- Massive data processing via network
- Linking with image processing and image recognition technologies and speed increase
- Improvement of memory bandwidth usage by high-resolution cameras and high-definition monitors

To resolve these issues, simply waiting for the performance of hardware to improve is not sufficient; improving higher-performance basic graphics libraries based on a full knowledge of hardware and drivers is also necessary. At present, we are working to offer these as user-friendly APIs.

7. Conclusion

This paper has presented the architecture of graphics software that runs on the MB86R11 evaluation board and examples of application.

In the future, multicore and multi-GPU configurations will be the mainstream of graphics SoCs and applications will be run at higher degrees of parallelism. This is expected to further sophisticate linking between applications and use of human machine interface (HMI) including 3D graphics. In such system environment, we intend to develop software capable of maximizing the performance of graphics SoCs and provide a development environment that facilitates the development of applications.

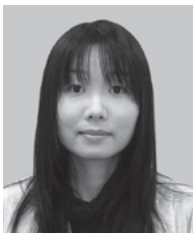
References

1) The Khronos Group. *http://www.khronos.org/*
2) Fujitsu Semiconductor: Graphics Solutions.

http://www.fujitsu.com/emea/services/microelectronics/gdc/
3) Fujitsu Laboratories: Wraparound 3D Monitor System. *http://www.fujitsu.com/global/news/pr/archives/month/2008/20081117-01.html*
4) S. Shimizu et al.: Wraparound View System for Motor Vehicles. *Fujitsu Sci. Tech. J.*, Vol. 46, No. 1, pp. 95-102 (2010).
5) Fujitsu : The World's first 360° Wrap-around View System for automotive applications. *http://www.fujitsu.com/downloads/MICRO/fme/documentation/g08.pdf*
6) Tech-On! SPECIAL: In-vehicle Graphics Solution (in Japanese). *http://special.nikkeibp.co.jp/ts/article/aaa0/112983/*

Yuya Tagami
Fujitsu Semiconductor Ltd.
Mr. Tagami is currently engaged in development of graphics-related software (OMNIVIEW Library).

Makoto Watanabe
Fujitsu Semiconductor Ltd.
Mr. Watanabe is currently engaged in development of graphics-related software (OMNIVIEW Library).



Yuko Yamaguchi
Fujitsu Semiconductor Ltd.
Ms. Yamaguchi is currently engaged in development of graphics-related software (OMNIVIEW Library).