Introduction

Ten years after the release of the first OpenGL® safety certifiable API specification the Khronos Group, the industry group responsible for graphics and video open standards, has released a new OpenGL safety certifiable specification, OpenGL SC 2.0.

While both OpenGL SC 2.0 and the earlier safety critical OpenGL specification, OpenGL SC 1.0.1, provide high performance graphics interfaces for creating visualization and user interfaces, OpenGL SC 2.0 enables a higher degree of application capability with new levels of performance and control.

This white paper will introduce OpenGL SC 2.0 by providing context in relationship to existing embedded OpenGL specifications, the differences compared to the earlier safety certifiable OpenGL specification, OpenGL SC 1.0.1 and an overview of new capabilities.

Overview of OpenGL Specifications

OpenGL Graphics Language (the GL in OpenGL) Application Programming Interfaces (API) is an open specification for developing 2D and 3D graphics software applications. There are three main profiles of the OpenGL specifications.

- OpenGL – desktop
- OpenGL ES – Embedded Systems
- OpenGL SC – Safety Critical

From time to time the desktop OpenGL specification was updated to incorporate new extensions which are widely supported by Graphics Processing Unit (GPU) vendors reflecting new GPU capabilities. The embedded systems OpenGL specification branches from the desktop OpenGL specification less frequently and is tailored to embedded system needs. The safety critical OpenGL specification branches from the embedded systems OpenGL specifications and is further tailored for safety critical embedded systems.

Figure 1 shows the high level relationship between the OpenGL specifications providing cross-platform APIs for full-function 2D and 3D graphics on embedded systems.

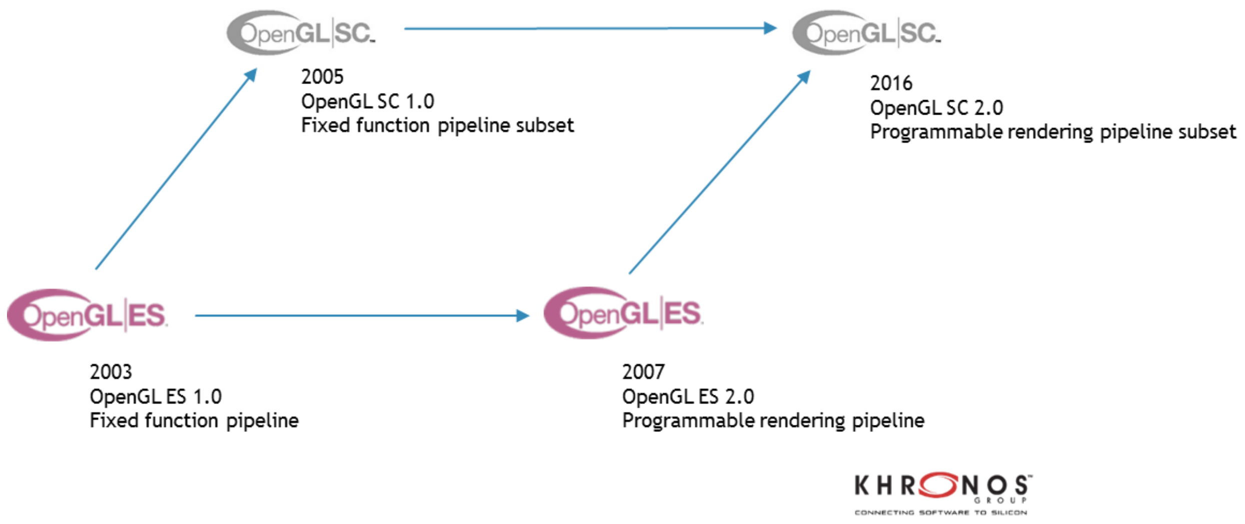

Figure 1: Relationship between the OpenGL specifications for Embedded Systems

OpenGL ES 1.x and OpenGL SC 1.0.1

OpenGL ES 1.0, was based on OpenGL 1.5 with some APIs removed and fixed point integer support added. OpenGL SC 1.0.1 removes functionality from OpenGL ES 1.0 to minimize implementation and safety certification costs. It also adds functionality, such as display lists, that are required to support legacy and auto-generated display applications in safety critical markets. In the end, OpenGL SC 1.0.1 is similar to, and is specified as a delta specification to, OpenGL 1.3. These specifications are defined relative to available GPU hardware implementations at that time which were fixed function pipeline architectures.

OpenGL ES 2.0 and OpenGL SC 2.0

OpenGL ES 2.0 combines an OpenGL Shading Language (GLSL) with an OpenGL API based on OpenGL ES 1 with some of the fixed functionality removed to minimize the cost and power consumption of advanced programmable graphics systems. OpenGL ES 2.0 is defined relative to OpenGL 2.0 to support GPU implementations with programmable pipeline architectures. The fixed functionality that was removed can be easily be replaced by shader programs.

The OpenGL SC 2.0 specification defines a safety critical subset of OpenGL ES 2.0. In addition, portions of the GL EXT texture storage and GL KHR robustness extensions have been adopted as part of the core OpenGL SC 2.0 specification.

Key Differences between OpenGL SC 1.0.1 and OpenGL SC 2.0

The significant differences between OpenGL SC 1.0.1 and OpenGL SC 2.0 are the addition of programmable shaders and the removal of some fixed functionality which 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 Figure 2.

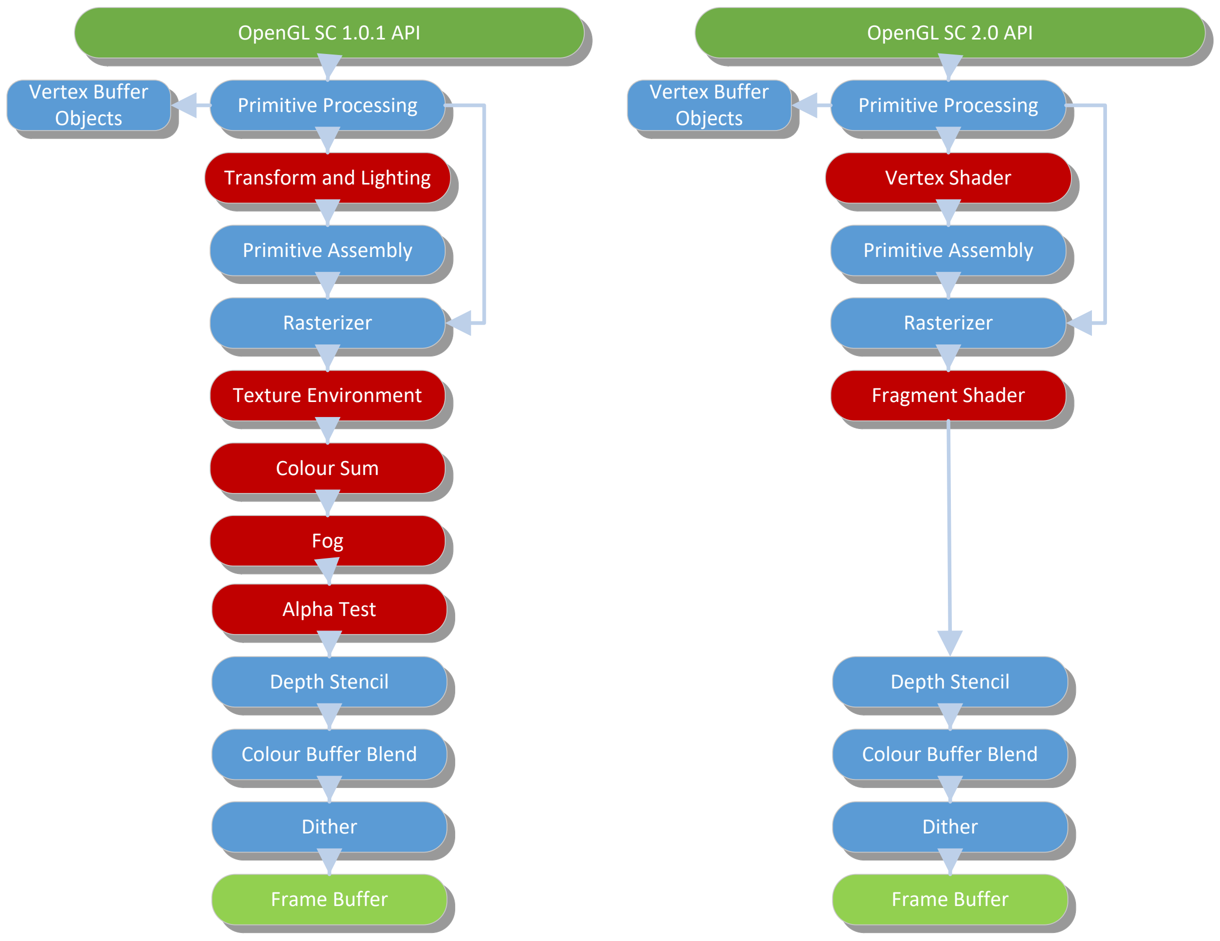

Figure 2: Differences Between Fixed Graphics Pipeline and Programmable Graphics Pipeline

As one can see, it is a 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 and the like yourself using GLSL vertex and fragment shaders.

Introduction to Shaders

Shaders offer a considerable number of advantages to your application:

- Shaders give you simplicity by allowing you to write code that expresses an algorithm directly compared to complex blocks of configuration calls
- Shaders allow for longer and more complex algorithms to be implemented using a single rendering pass
- Your application can switch between different shaders with a single function call compared to the overhead with reconfiguring the fixed-function pipeline.

There is an execution difference between OpenGL and shaders where program control remains with the CPU for OpenGL and program control resides within the GPU for shaders. OpenGL commands are parsed by a Central Processing Unit (CPU) which results in commands being send to the GPU to be acted upon by the GPU. Shader programs are transferred to the GPU and executed within the GPU within an array of Single Instruction Multiple Data (SIMD) processors.

There are two common types of shaders, vertex shaders and Fragment shaders.

- **Vertex Shaders**

  A vertex shader is an array of operations that are meant to occur on each vertex that is processed. The language used for vertex shaders is described in the OpenGL Shading Language Specification.

  The vertex shader is executed once per vertex in a scene to transform 3D position in virtual space to the 2D coordinate at which it appears on the screen. Vertex shaders can also manipulate position, colour and texture coordinates.

- Fragment Shaders

  A fragment shader, sometimes referred to as a pixel shader, is an array of operations that are meant to occur on each fragment that results from rasterizing a point, line segment, polygon, pixel rectangle or bitmap. The language used for fragment shaders is described in the OpenGL Shading Language Specification.

  The fragment shader is executed once per pixel per geometric primitive to calculate colour of individual pixels along with lighting, texturing bump mapping etc.

OpenGL Shading Language

The OpenGL Shading Language (GLSL) is used for programming vertex and fragment shaders, the programmable portion of the graphics pipeline, and is specified separately from the OpenGL language specification. GLSL is based on the ANSI C programming language extended for use with 3D graphics.

The OpenGL ES Shading Language (also known as GLSL ES or just ESSL), which is defined for OpenGL ES 2.0 is applicable to OpenGL SC 2.0 with limitations as documented in the OpenGL SC 2.0 specification. While it is based on GLSL, it is not the same as the two APIs have progressed at different rates. Therefore, you may use desktop GLSL shaders as a reference however you should not expect them to work directly with OpenGL SC 2.0. A better source of reference shaders is from the widely supported embedded device space where shaders have been used to provide feature rich applications since 2007.

While shader compilers are integrated into commercial desktop OpenGL drivers, for embedded systems using OpenGL ES 2.0 and OpenGL SC 2.0 the shaders are compiled and linked off-line into a program binary object which is loaded into the GPU for execution at runtime.

Should You Be Using OpenGL SC 2.0?

Both OpenGL SC 1.0.1 and OpenGL SC 2.0 have a number of characteristics in common including:

- **Safety Certifiable** - Both OpenGL SC 1.0.1 and OpenGL SC 2.0 drivers are available with evidence to support the highest safety certification levels such at FAA DO-178C Design Assurance Level (DAL) A, EASA ED-12C DAL A, ISO-26262 ASIL D amongst others.
- **GPUs** – Both OpenGL SC 1.0.1 and OpenGL SC 2.0 drivers are supported and available for the popular embedded GPUs from AMD, Intel, Verisilicon and NXP.
- **Operating Systems** – Both OpenGL SC 1.0.1 and OpenGL SC 2.0 drivers are supported on operating systems from DDCi, Green Hills Software, Lynx Software, SYSGO and Wind River as well as custom operating systems.
- **Tools** – Both OpenGL SC 1.0.1 and OpenGL SC 2.0 are supported by leading Human Machine Interface (HMI) tool vendors.
- **FACE** – With the release of Future Airborne Capability Environment (FACE) version 3.0, there is a choice of using OpenGL SC 1.0.1 or OpenGL SC 2.0 for the safety profile.

Essentially moving from OpenGL SC 1.0.1 to OpenGL SC 2.0 does not involve giving anything up, other than having built-in convenience functions that make simple 3-D applications easier. What OpenGL SC 2.0 does is open the possibilities for enhancing and providing new capabilities through:

- **Performance** – While you are probably going to see identical performance using both OpenGL SC 1.0.1 and OpenGL SC 2.0 if you create shaders that just simulate the OpenGL SC 1.0.1 fixed function pipeline in OpenGL SC 2.0. Performance gains can be achieved with OpenGL SC 2.0 through
    - Performance enhancements in custom shaders to do only the specific function(s) required rather than more general fixed functions
    - Rendering lots of objects
- **Control** - OpenGL SC 2.0 provide a higher degree of control by providing a fully programmable pipeline through the use of shaders to take advantage of current GPU architectures. This means you can write your own custom shaders tailored for dealing with your geometry and textures and how they are displayed to the screen. That is OpenGL SC 2.0 enables you do things that would be difficult or even not possible to do in OpenGL SC 1.0.1 and make for better looking and faster effects.

It is these new possibilities for graphics capabilities leveraging the widespread use of shaders from the embedded markets that will bring competitive advantages through enhanced and new capabilities that were not considered possible or cost effective previously for safety critical applications.

Where to Find OpenGL SC Drivers and Additional Information

In addition to CoreAVI's high Technology Readiness Level (TRL) OpenGL SC 1.0.1 drivers with certification evidence, CoreAVI demonstrated the industry's first OpenGL SC 2.0 driver on April 20, 2016. CoreAVI is also formally in the process of certifying its OpenGL SC 2.0 driver with several major avionics display system manufacturers that are requiring FAA DO-178C Level A certification.

More information about CoreAVI's ArgusCore SC1, OpenGL SC 1.0.1 API with extensions, and ArgusCore SC2, OpenGL SC 2.0 API with extensions, certifiable drivers along with a comparison driver features can found here: CoreAVI safety certifiable graphics drivers.

Contact CoreAVI to find out what we are working on and to discuss your demonstration/evaluation requirements: Sales@CoreAVI.com

OpenGL SC specifications and further information can be found on the Khronos website here: Khronos OpenGL SC Official Site