GENIVI®

TECHNOLOGY BRIEF

NOVEMBER 2018

Category: Graphics Sharing and Distributed HMI

Distributed Graphics Control Through API Remoting

Summary

Five primary techniques for distributing graphics and HMI between cooperating systems have been identified in the GENIVI Graphics Sharing and Distributed HMI project (GSHA).

This brief describes the API Remoting method.

Key Characteristics

- Bandwidth efficient (for certain cases)
- Offloads rendering effort to receiver
- Flexible, while the controller has detailed control over results
- Potential API complexity – needing synchronized software/API updates

Description

In the context of graphics sharing, API Remoting takes an existing graphics API and makes it accessible through a network protocol, thus enabling one system to effectively call the graphics/drawing API of another system.

Sometimes an existing API of a local graphics library is almost directly extended to the network, turning it into an RPC (Remote Procedure Call) interface. There are many technologies for making APIs independent of the location of the requesting and receiving software components. The "Generic Protocol Evaluation" project in GENIVI deals with analyzing those methods.

In other cases, new custom programming interfaces are created, specially tailored for distributed graphics.

Fundamentally, we are dealing with system A remotely invoking drawing commands on system B so that the actual drawing/rendering is done on the receiving system. This approach is different from Surface Sharing where the drawing/rendering is first completed on system A, and then a bitmap image is transferred. A 3D-scene can be quite complex and consist of different objects, textures, animations and relationships between them - so an efficient remote rendering API should reflect that.

Design Considerations

Various aspects should be considered when choosing the graphics sharing approach:

- Network bandwidth
- Latency, affecting changes and animations
- Processing requirements on system A and system B, and associated required hardware resources
- Complexity, compatibility, API versions and software upgrades.

Bandwidth usage differs depending on if the solution sends drawing commands (remoting) or images (surface sharing). Thus advantages of API remoting will depend on the rate of change in large textures and consequently if these are transferred mostly once initially, or repeatedly. As noted, the graphics processing itself is primarily shifted to the receiving system B. Remember that a large API between systems likely correlates with integration effort and it might require more frequent synchronized updates, keeping the controller and receiver compatible. Please refer to the full whitepaper (under development) for additional analysis of such trade-offs.

Handheld Devices

Smartphones should be considered part of the extended HMI of the car. Simple data-exchange like Shared State is often appropriate but it's also possible that the car more directly controls the graphics shown on the mobile device. The RAMSES framework is cross-platform and has been tested on Android to this effect.

Case Study: RAMSES

RAMSES is a framework for defining, storing and distributing 3D scenes for HMIs. From a user's perspective, RAMSES looks like a thin C++ scene abstraction on top of OpenGL. RAMSES supports higher-level relationships between objects such as a hierarchical scene graph, content grouping, off-screen rendering, animations, text, and compositing. RAMSES is primarily categorized as API remoting. Commands can be sent to another system (that also exchange state in the RAMSES case) and the actual rendering is executed at the receiving side.

RAMSES distinguishes between content creation and content distribution; the same content (created with RAMSES) can be rendered locally and on one or more network nodes. RAMSES handles the distribution of the scenes, and offers an interface to control the final composition - which scene to be sent where, shown at which time, and how multiple scenes and/or videos will interact together. The control itself is subject to application logic (HMI framework, compositor, smartphone app, etc.).

Usage in Virtualized Environments

Since API Remoting deals simply with commands sent over a network it should work using a virtual network provided between virtual machines with minimum adaption. It is possible that some particular shared memory optimizations can be added for graphics buffers or simply in the generic networking code.

Conclusions

API Remoting is a method to have graphical interaction that gives the control of the graphics to one system while offloading the graphics rendering effort to the other. Bandwidth efficiency can be better than alternatives, depending on the situation. A potentially large interaction interface may increase the need for synchronized software updates for sending and receiving systems.

References & Additional Reading

- *https://github.com/GENIVI/ramses*
- Graphics Sharing & Distributed HMI Whitepaper (in development)

Authors

Violin Yanev (Violin.Yanev@bmw.de), BMW
Participants of the Graphics Sharing / Distributed HMI working group