From Separated ECUs to a Display Cluster

April 18, 2018

Jonathan Conrad, Violin Yanev, Bernhard Kißlinger
BMW Car IT
Challenge

- Multiple displays
- Different hardware, different companies
- Seamless integration of content
  - Content not fixed to one display
Agenda

1. Challenge
2. Solution Ideas
3. RAMSES Concepts & Features
4. Live Demonstration
5. Wrap-up
Solution ideas
Solution ideas:

1. One ECU with multiple displays

- No solution for distribution necessary
- No network issues
- High computation power needed, scaling to more displays problematic
- Interaction between content from different processes limited

ECU 1

<table>
<thead>
<tr>
<th>Display 1</th>
<th>Display 2</th>
</tr>
</thead>
</table>

`glBindFramebuffer(…)` `glClear(…)` `glClearColor(…)`...
`glUseProgram(…)` `glBindBuffer(…)` `glActiveTexture(…)` `glUniform1f(…)` `glDrawElements(…)`
Solution ideas:

2. Video distribution

- Easy integration of existing applications
- High computation power needed
- High bandwidth requirements
- Compression artifacts possible
- Availability of hardware encoders and decoders can limit deployment
- Interaction between content from different sources limited
Solution ideas:
3. OpenGL commands streaming

- Easy integration of OpenGL-based applications
- No compression artifacts
- Easier scaling to higher resolutions
- No GPU needed on sending side
- Limited to OpenGL-based applications
- Medium bandwidth requirements (full description for each single frame has to be transferred)
- Platform-dependencies with receiving side
- Interaction between content from different sources complex
Solution ideas:
4. Scene-based distribution

- Low network bandwidth needed especially after initial transfer
- No compression artifacts
- Easier scaling to higher resolutions
- No GPU needed on sending side
- Graphical interaction possible between scenes from different ECUs

Application has to provide content with special API

```
glBindFramebuffer(.
) glClear(.
) glClear(.
)...
) glUseProgram(.
) glBindBuffer(.
) glActiveTexture(.
) glUniformf(.
) glDrawElements(.
)
```
Update of frames: Video distribution

Frame A

```c
// OpenGL commands
glBindFramebuffer(…)
glClearColor(…)
glClear(…)
…
glUseProgram(…)
glBindBuffer(…)
glActiveTexture(…)
glUniform1f(x)
glDrawElements(…)
…
```

'Full new image' (maybe compressed)

Frame B

```c
// OpenGL commands
glBindFramebuffer(…)
glClearColor(…)
glClear(…)
…
glUseProgram(…)
glBindBuffer(…)
glActiveTexture(…)
glUniform1f(y)
glDrawElements(…)
…
```

Pixels/Video

April 18, 2018
Update of frames: OpenGL commands streaming

Frame A

```
glBindFramebuffer( ... )
glClear( )
glClearColor( )
... 
glUseProgram( )
glBindBuffer( )
glActiveTexture( )
glUniform4fv( )
gDrawElements( )
... 
```

Frame B

```
glBindFramebuffer( ... )
glClearColor( )
glClear( )
... 
glUseProgram( )
glBindBuffer( )
glActiveTexture( )
glUniform1f( )
gDrawElements( )
... 
```

Slightly changed list of commands

OpenGL command stream
Update of frames: Scene-based distribution

Frame A

```c
glBindFramebuffer(...)  
glClearColor(...)  
glClear(...)  
...  
glUseProgram(...)  
glBindBuffer(...)  
glActiveTexture(...)  
glUniform1f(...)  
glDrawElements(...)  
...  
```

Frame B

```c
...  
glUseProgram(...)  
glBindBuffer(...)  
glActiveTexture(...)  
glUniform1f(y)  
...  
```

Scene + state

changeValue $\rightarrow y$
Update of frames

Frame A

changeValue -> y

Slightly changed list of commands

'Full new image' (maybe compressed)

Frame B

Scene + state

OpenGL command stream

Pixels/Video

12 | April 18, 2018 | Copyright © GENIVI Alliance 2018
RAMSES Concepts & Features
RAMSES Software Stack

- Business Logic/App
- Widget Framework
- RAMSES Client
- RAMSES Renderer
- OpenGL
- GPU

Content provider
Content consumer
RAMSES Software Stack

Communication middlewares: SomeIP (abstraction for two different stacks), custom TCP communication
RAMSES scenes

- RAMSES works with scenes
- A scene == content which belongs together
- For example, a radio application could have two scenes:
  - Scene which has the radio’s own UI
  - Scene which shows the list of all songs (targeted for display on different ECU)
RAMSES scenes compared to OpenGL

- Converting OpenGL to RAMSES content is mostly easy
- Most OpenGL constructs have a RAMSES counterpart, e.g.:
  - `glDrawElements()` ~ `ramses::MeshNode`
  - `glCreateProgram()` + `glCompileShader()` ~ `ramses::Effect`
  - `glBindFramebuffer()` ~ `ramses::RenderTarget`

- Difference:
  - OpenGL’s frame is continuously “recreated” – even with small changes
  - RAMSES objects lifecycle is not per-frame
  - Selective changes possible, can change individual objects or groups
RAMSES offers additional features on top of OpenGL that help to reduce data bandwidth

For example, have a scene graph instead of list of draw commands:

Such optimizations benefit remote and local scenes
Interaction between scenes

- Independent scenes can exchange data via RAMSES
  - Any “uniform” or “constant” data – colors, animated values, etc.
  - Textures
  - Positions

- Example with color:
Further features

• Cross platform:
  – Windows, Linux, Integrity OS
  – Wayland, X11, WGL, Integrity OS window system
  – Desktop OpenGL (4.2, 4.5)
  – Embedded OpenGL (ES 3.0+)
  – Clang, GCC, MSVC, Integrity OS compiler

• Wayland support with nested compositing

• Text rendering

• Animations

• Content authoring tool: RAMSES Studio
Live Demonstration
Demonstrator setup

- Instrument Cluster UI
- Central UI + List scene
- City model
- Car application

Ethernet

PC 1 (Linux)

PC 2 (Windows)

- All code/rendering is live with RAMSES
- Each application is own process
Benchmarks by LG Electronics

Compare the Performance of Radio List App

CPU Load

% 60 50 40 30 20 10 0
1 3 5 7 9 11 13 15 17 19

- Radiolist
- Ramses-Renderer
- screen-sender
- screen-receiver

every 3 seconds sampling

50.3%
21.8%
11.32%
0.37%

Bandwidth

Kb 2000
1500
1000
500

3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33

~244X 950Kb/s

Ramses
ScreenCapture

GPU Load

% 90 80 70 60 50 40 30 20 10 0
1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91 97

- Radiolist
- Ramses-Renderer
- screen-sender
- screen-receiver

75.3%
51.7%
25.1%
0%

Please visit LG Electronics booth for full benchmark
Benchmarks by LG Electronics

Compare the Performance of Radio List App

Receiver: Video transfer (H.264)

Receiver: RAMSES transfer
Benchmarks by LG Electronics

Compare the Performance of Radio List App

Video transfer (H.264)

RAMSES transfer
Benchmarks by LG Electronics (2)

Compare the Performance of Navigation App

- Video transfer (H.264)
- RAMSES transfer
Wrap-up

Applications (or underlying Widget framework) must be adapted to use RAMSES API

More interaction of content than video allows seamless UI
Graphical flexibility
Low bandwidth

Will be open sourced in Q3 2018!
Questions?

Thank you!

Visit GENIVI at http://www.genivi.org or http://projects.genivi.org
Contact us: help@genivi.org

GENIVI is a registered trademark of the GENIVI Alliance in the USA and other countries.
Copyright © GENIVI Alliance 2018.