From Separated ECUs to a Display Cluster
Oct 11, 2018

Violin Yanev, Vaclav Kyba
BMW Group
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. Hands-on workshop
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

![Diagram of ECU with multiple displays](image)
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
Update of frames: Video distribution

Frame A

- glBindFramebuffer(…)
- glClearColor(…)
- glClear(…)
- ...  
- glUseProgram(…)
- glBindBuffer(…)
- glActiveTexture(…)
- glUniform1f(…)
- glDrawElements(…)

→ ‘Full new image’ (maybe compressed)

Frame B

- glBindFramebuffer(…)
- glClearColor(…)
- glClear(…)
- ...  
- glUseProgram(…)
- glBindBuffer(…)
- glActiveTexture(…)
- glUniform1f(…)
- glDrawElements(…)

→ Pixels/Video

Copyright © GENIVI Alliance 2018
Update of frames: OpenGL commands streaming

Frame A

OpenGL command stream

Frame B

Slightly changed list of commands
Update of frames: Scene-based distribution

Frame A

Frame B

Scene + state

changeValue -> y

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

glBindFramebuffer(...) 
glClearColor(...) 
glClear(...) 
... 
glUseProgram(...) 
glBindBuffer(...) 
glActiveTexture(...) 
glUniform1f(gy) 
glDrawElements(...) 
... 

11 | Oct 11, 2018 | Copyright © GENIVI Alliance 2018
Update of frames

Frame A

- `glBindFrameBuffer(..)`
- `glClearColor(..)`
- `glClear(..)`
- ...
- `glUseProgram(..)`
- `glBindBuffer(..)`
- `glActiveTexture(..)`
- `glUniform4f(x)`
- `glDrawElements(..)`
- ...

Slightly changed list of commands

- `changeValue -> y`

'Full new image' (maybe compressed)

Frame B

- `glBindFrameBuffer(..)`
- `glClearColor(..)`
- `glClear(..)`
- ...
- `glUseProgram(..)`
- `glBindBuffer(..)`
- `glActiveTexture(..)`
- `glUniform4f(y)`
- `glDrawElements(..)`
- ...

Scene + state

OpenGL command stream

Pixels/Video
RAMSES Concepts & Features
RAMSES Software Stack

Communication middleware

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

Content provider

Communication middleware

Content consumer

RAMSES Renderer
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)
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 scenes compared to OpenGL

- 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:

  ![Scene Graph Example]

  - Not optimized
  - Optimized

- 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:
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

- All code/rendering is live with RAMSES
- Each application is own process

PC 1 (Linux)

PC 2 (Windows)
Benchmarks by LG Electronics

Compare the Performance of Radio List App

**CPU Load**
- RadioList: 50.3%
- Ramses-Renceller: 21.8%
- screen-sender: 11.32%
- screen-receiver: 0.37%

Every 3 seconds sampling

**GPU Load**
- RadioList: 75.3%
- Ramses-Renceller: 51.7%
- screen-sender: 25.1%
- screen-receiver: 0%

**Bandwidth**
- RadioList: ~244X
  - Ramses: 950Kb/s
  - ScreenCapture: 25.7Kb/s
Benchmarks by LG Electronics

Compare the Performance of Radio List App

Receiver: Video transfer (H.264)
Receiver: RAMSES transfer

every 3 seconds sampling
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)**
  - ~3.8X
  - 210 MB/s

- **RAMSES transfer**
  - 5.45 MB/s
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
Questions?

Thank you!

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

Picture sources:
https://www.press.bmwgroup.com/deutschland/photo/compilation/T0274140DE/bmw-concept-x7-iperformance-eine-neue-grosszueigigkeit

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