The instrument clusters of today's cars show a clear trend away from electromechanical pointer instruments and toward solutions with significant focus on graphical displays. This paper provides an overview of automotive instrument clusters with advanced graphic capabilities - today and in the foreseeable future.

In this emerging field there is still a quick evolution ongoing driven by customer expectations influenced by graphical solutions from the consumer world. Bringing those expectations together with automotive quality requirements at affordable cost is still a challenge that has to be addressed by new methodologies and tools.

In this paper the discussion of instrument clusters is split into several topics: the use-cases, graphical techniques, technical implementations and design decisions taken in such complex embedded systems. As there are many techniques adopted from the PC arena a quick overview of selected areas is presented and its adoptions for the embedded world are outlined.

A focus is put on the creation of the graphical content and human-machine interfaces and the evolving tools support in that area. This paper shows several approaches and solutions for a smooth development flow starting from HMI graphics development down to optimized porting on embedded graphic accelerators.

While current designs focus on imitating classical pointer instrument clusters, this paper will show some upcoming trends and possibilities in the design of advanced user interfaces.

1 Introduction
In recent years the traditional mechanical automotive instrument cluster has been more and more extended by electronic displays. The first step in the evolution was the introduction of liquid crystal displays used for mileage information and other simple alpha-numeric information.
The reduced prices of TFT displays and the availability of the required compute performance in the embedded space now allow affordable graphic based solutions. As the instrument cluster is one of the most prominent parts of the HMI and very evident to the driver it is an important differentiation feature for the car manufacturers. For high-end cars this leads to the situation that manufacturers are significantly increasing the level of investment into that area. Currently TFT displays in that area still seem to be considered an added value in itself by the driver as it is considered a high-tech feature.

Since the "gauges with moving pointers" visual paradigm is well accepted by consumers and can very quickly communicate both speed and acceleration to the driver, it is necessary to simulate the mechanical instruments using advanced graphic technology. Exploitation of additional possibilities is just at the beginning, although there is a huge potential for improvements in that area. An obvious example is the reconfigurability of the graphics based dashboard which is a key advantage. The display content can be changed based on the context, which is not the case with mechanical gauges. Looking at consumer industry products it is obvious that well defined user interfaces are a key differentiation factor already today.

The general idea behind any optimization in that area is to display more complex information in a simpler way: the ways to achieve this maxim for the automotive cluster is currently an evolving area. The advantages are not solely technical, there are also commercial aspects. The graphic technology easily allows adding branding to the dashboard. Potentially it could even open new revenue streams by allowing the use of skins for the HMI or enabling the download of additional applications.

Automotive dashboards with graphic displays impose new technical challenges on the development of such solutions. The level of complexity is increased by at least one order of magnitude. While the graphic technology used in that area itself is adopted from the PC arena its application in the car is new. This requires new skills from the development teams and quite some time to go through a steep learning curve.

Consumers are used to PC graphic quality, but the fact that the compute performance in PCs is well beyond what is available in the embedded space makes rendering close to photo-realistic images even more challenging. There are also very specific use-cases in a dashboard which a PC does not have to deal with, such as animating a gauge needle - typically 60fps animation is needed to make this look smooth, whereas video games on a PC only need to run at 30fps to look good.

In contrast to the PC arena there are additional challenges concerning power consumption and
temperature - some of those challenges are similar to those in the handheld devices.

A key feature of the automotive solution is functional safety which is not required in the computer and handheld markets. At least a part of the information presented in the automotive instrument cluster is considered safety relevant. Presenting wrong information is not acceptable here.

Use Cases

2 Use Cases
In the current market we see a clear segmentation into several classes of graphical cluster displays. The segmentation is basically driven by the cost of the solutions as the price adder for larger display is still significant and not affordable for mid- to low-end cars. What is common for all segments is the need for analog needles – implemented either as electromechanical needles or rendered by a sophisticated graphic solution in photo-realistic quality.

2.1 Center Add-on Display
An "Add-on display" is the most common use-case that is currently becoming standard even in low-end cars. The cluster display still has the mechanical needles but features an additional display that is usually located in the center between the gauges. Entry level uses ~4 inch QVGA or WQVGA displays.

The display is used to present additional information like current fuel consumption, temperature or similar nice to have info. Optionally information from the infotainment system is presented here.

Figure 2: Audi A4 (2009) Instrument Cluster.

In upper-mid to high-end cars larger add-on displays are common. This allows the display to be used for camera images like e.g. night vision, rear-view camera or even a bird-view park-assist image.

Due to the electromechanical instruments the partitioning of the cluster is fixed and cannot be adapted to the situation.

2.2 Two Displays with central gauge
A compromise still based on a traditional electromechanical needle but offering extended graphical capabilities is a cluster with two displays on both sides of a central mechanical gauge.
It offers significantly more flexibility in displaying information but still has the fixed setup with a fixed electromechanical needle. Screen resolutions common for this use-case are up to SVGA per display.

2.3 Fully Configurable Cluster
For fully configurable clusters large displays with resolution of up to 1280x480 pixels are used today with even higher resolutions planned in future. Those displays are still a significant cost factor which currently limits its application to high-end cars. As there is no mechanical needle available current implementations spend quite some effort to render photorealistic analog needles.

Since the cluster content is fully software defined it can be flexibly adapted to the situation. This allows efficient integration of additional situation-dependent information like night vision into the cluster by resizing or moving content that is not that relevant in the current situation.

2.4 Head-up Display
Head-up displays are projecting the graphic information on the windscreen. The fundamental advantage of those displays is that the information is presented in the normal field of view of the driver looking onto the street. Usually the head-up display is projected in a way that the virtual distance of the graphic is several meters in front of the car which doesn’t even require refocusing the eyes when changing from the world outside to the head-up display information.

While head-up displays used to have a rather low resolution and limited color depth, if not even monochrome, this changed fundamentally in the recent past. Meanwhile even for head-up displays rather complex graphic content and a resolution up to WVGA at true color resolution is required.
The content of head-up displays has to be pre-warped to compensate the distortion introduced by the curved windscreen. Today pre-warping is in most cases implemented using a curved mirror which compensates the distortion introduced by the windscreen.

A cost efficient and more flexible solution is electronic warping. There the pre-warping is done on the graphic data using the graphic accelerator or dedicated hardware. Electronic dewarping introduces the flexibility to change the de-warping function on the fly. This enables e.g. to adapt the position of the head-up display depending on the size and position of the driver.

Head-up displays are usually used in combination with a TFT display in the instrument cluster.

**Techniques**

3 Techniques
The use-cases are very different depending on their complexity. This is caused by several factors.

- **Screen size**: the big variation of screen size leads to a significant difference in the number of pixels that have to be generated. In solutions currently in development or on the market 75kPixel up to more than 1MPixels have to be handled for a frame.
- **Animation frequency**: As soon as fast motion is displayed it is extremely important to provide a high animation frequency. An example for a critical case is a gauge needle for a speedometer or rpm meter. For a needle an animation frequency of at least 60fps is required to get a decent visual impression.
- **Scene complexity**: the variation of the expected optical effects ranges from typical graphical user interface menus up to rendered 3D scenes with lighting, reflections and shadows.
To tackle the drastically varying complexity of the use-cases outlined in the previous chapter it is necessary to apply different techniques to generate the graphic.

Applying several graphic techniques simultaneously is a beneficial approach especially for the fully configurable cluster. This allows loading several graphic processing units in parallel while using the appropriate technique for each graphic element. Using several graphic processors in parallel even allows implementing different animation frequencies for the graphic elements depending on their specific requirements. This prevents the common problem that the entire scene has to be rendered at 60fps due to needles imposing this requirement.

3.1 Raster Graphics

Figure 6: Raster Graphics Example.

In raster graphics, visual elements are represented by the color intensity values of all pixels belonging to the element. A key attribute of raster graphic is that it is resolution dependent. Zooming leads to significant optical degradation.

Processing of graphical elements usually handles a rectangular area of pixels.

Raster graphics is a common technique for natural images (photos), most graphic formats represent raster graphic (jpg, bmp, png, gif). There are many graphical design tools which deal with raster graphics (Adobe Photoshop, GIMP, Aperture).

3.1.1 Raster Graphics Processor

To accelerate raster graphics the standard solution is a raster graphic processor which is able to copy / fill / combine rectangular areas of pixels. A raster graphic accelerator basically performs a memory to memory operation, i.e. it reads the source data from memory and writes the result data back to memory.

There is no commonly accepted API Standard available. Several proprietary APIs provide similar functionality.

More sophisticated raster graphics accelerators also offer the ability to draw graphic primitives like lines and circles.

3.1.2 Direct Blitting Engine (Sprite Engine)
Similar to a raster graphics processor a direct blitting engine handles rectangular areas of pixels. The key difference is the fact that the result of the graphical operation is not written back to memory. The direct blitting engine is part of the display controller which assembles the final image for every frame reading from various locations in memory.

The key advantage of a direct blitting engine for embedded systems is the saving of memory and memory bandwidth. On top of that it is very efficient to generate the graphic as only metadata like the position of a certain rectangle has to be modified and not the pixel data itself.

The biggest disadvantages of a direct blitting engine are the incorporated limitations. As soon as the capabilities of the unit are exceeded it becomes hard to generate more complex graphics. When using a raster graphic processor this only leads to increased rendering time. Solutions in that field have a proprietary API, there are significant differences concerning the functionality of the engines.

3.2 Vector Graphics

![Vector Graphics Example](image)

Vector graphics is based on a resolution independent, mathematical description of shapes in 2D space. It is therefore possible to render it in any resolution achieving perfect quality.

Vector graphics is a common technique which is used in many applications today. The most prominent example is the TrueType font, but it is also used in the Scalable Vector Graphics format SVG - understood by any up-to-date browser. Formats like Adobe Flash also heavily depend on vector graphics. There are several applications available to generate vector graphics: Adobe Illustrator, Inkscape, Adobe Flash Professional.

For vector graphic accelerators an industry standard API is available – OpenVG[0].

Latest trend shows increasing support for OpenVG offered by several HMI tool vendors in the
3.2.1 OpenVG Processing Unit
There are meanwhile several embedded devices available in the automotive market directly accelerating the functions of the OpenVG API. This provides high rendering performance while achieving very high quality results.

3.2.2 OpenVG Using a 3D Graphic Processing Unit
Several vendors of 3D graphic processing unit offer special drivers implementing OpenVG using a 3D graphic processing unit. The fundamental problem of this approach is the need to divide the vector graphic elements into triangles very early in the processing pipeline. The mature 3D graphic processing units usually do the rendering at a reasonable speed but suffer optical degradations in critical areas of the image due to the need to tessellate the graphical objects into triangles. The result is that the rendered image has lower visual quality compared to an image rendered by a dedicated OpenVG GPU.

3.2.3 OpenVG Software Renderer
Pure software implementations of the OpenVG API achieve a poor rendering speed while achieving a decent optical result. For automotive instrument clusters this approach is currently too slow to be practical.

3.3 3D Graphics

Real-time 3D rendering today is based on a 3D model consisting of triangles. To achieve decent results textures are applied to the triangles. The division into triangles defines the level of detail for the model and therefore has impact on the rendering quality. This makes the model to some extent resolution dependent.

For the embedded space there are three relevant API definitions for 3D graphics. They are derived from the OpenGL standard originally initiated by Silicon Graphics. For the embedded version several features not that relevant for embedded devices were removed. This leads to a reduced size of the hardware accelerators and simplifies the required drivers.
OpenGL ES 1.1[1]: API of the fixed function hardware. Fixed function indicates that the functionality and lighting model is defined and cannot be changed by the developer. This limitation lead to the situation that OpenGL ES 1.1 is no longer used for new projects.

OpenGL ES 2.0[2]: API of the programmable hardware. Many aspects of the rendering pipeline can be defined by so-called shaders. Shaders are dedicated optimized cores to handle the workload of the rendering process. The exact function of the shaders can be defined in a standardized way using a special language which is pretty close to C. This flexibility leads to significantly improved graphical results, indeed many common graphical effects are difficult or even impossible to achieve without this feature.

OpenGL ES 3.0[3]: evolutionary step from OpenGL ES 2.0 bringing significant functionality and portability enhancements. OpenGL ES3.0 is backward compatible to OpenGL ES 2.0 which allows a smooth transition. The specification was released recently; this standard is not yet used in automotive projects.

There are many applications available to generate 3D models. The most common ones are Autodesk 3ds Max, Blender, Autodesk Maya. OpenGL ES 2.0 is fully supported by the key tool vendors in the automotive space and clearly the standard way to implement sophisticated 3D graphic in automotive instrument clusters.

Development Flow

4 Development Flow
The development flow of automotive instrument cluster is currently still an evolving area. Depending on the underlying graphic technology different methodologies are used during this process. The process is split into several steps.

Usually it starts with an artistic design department delivering the graphic elements and drafts functionality and look of the entire solution. Starting from this point there are currently two different development paths used in the product development.

4.1 Manual-/Semi-Automated HMI Design Flow
In the manual or semi-automated HMI design flow the graphical elements are handed over to the software development department working on the program logic to actually implement the solution. As many details are often not fully defined this usually ends up in several time consuming iterations until the final results meet the expectation of the design department.

To shortcut those iteration cycles one potential solution employed for non-3D graphic is Adobe Flash - a graphical animation tool suited for pixel-accurate definition of animation sequences and human machine interfaces. The animation sequences are then converted to a format suitable for the embedded platform. This step can be supported by using a conversion tool. Another possibility is to hand-code the animations. Alternatively hand-coding can be supported by specific libraries covering the underlying functionality of Adobe Flash.

For the manual- / semi-automated design flow the control logic behind the HMI is usually implemented in a programming language like C or C++.

As there is no integrated continuous development flow, the synchronization between the graphical elements like animations and the underlying control logic has to be ensured manually. This makes changes in the HMI design difficult and error-prone especially when introduced late in the development process. On top of that there is significant effort required for documentation and
maintenance of such a hand-crafted solution. However this approach allows maximum optimization of the available resources and can be a good solution for systems that have relatively simple graphics or that have very tight cost or memory constraints.

4.2 Integrated HMI Design Flow
Enabled by the introduction of standard graphic APIs there are several solutions on the market today to streamline the development process and remove the necessity for hand-written specific code. Those products usually make use of graphical elements designed with the appropriate graphical design application and offer a graphical user interface to define the look and functionality of the instrument cluster's HMI. Using such HMI builders do not require programming skills. The work focuses on artistic aspects and the functionality of the generated HMI.

The output generated by a HMI development tool can be either a description file that is used by a runtime environment running on the embedded target. Another possibility is that the HMI development tool outputs source code for the target. This is then compiled for the target platform and usually linked together with a library providing a runtime environment.

For convenience there exist simulators to test the HMI directly on the development PC.

Some products offer also the possibility to check the consistency of the generated HMI and to generate documentation. Such features ease iterative development processes as well as last minute changes ensuring a high quality level. On top of that an up-to-date documentation is immediately available without extra effort.

HMI builders also have the potential to bridge the gap between the different graphic technologies by separating the actual representation from the HMI logic. This would allow generating simplified HMI on a low-end raster graphic platform and a fancy 3D HMI using the same HMI description. The re-use of the HMI control logic reduces the effort to implement and test this part of the HMI and ensure a consistent user interface between different product models.

A disadvantage of the integrated HMI design flow is the increased resource requirement. Solutions designed with such a tool usually require more memory and compute performance – both critical resources in the embedded space. Another drawback is the limited flexibility introduced by the standard elements of the design tool. This can often be overcome by designing a customer-specific widget class which has to be implemented in a programming language and imposes significant new effort for implementation, test, documentation and maintenance. An integrated solution commonly used in the automotive market is e.g., the Elektrobit HMI tool EB GUIDE.

State of the art HMI tools meanwhile even support solutions featuring multiple graphic processors to make optimal use of the compute performance offered.

As there are pros and cons for both design flows the decision to use a specific flow is always a tradeoff between hardware resources and development effort that has to be taken on a case-by-case basis driven by the cost target.

Discussion

5 Discussion
The graphic technology used in automotive instrument clusters offers plenty of new possibilities. Today most solutions focus on imitating the look and feel of mechanical instrument clusters. From the technological standpoint this is a challenging task that requires generating photo-realistic
images in real time. Especially looking at 3D graphics this puts significant load on dedicated 3D acceleration blocks and even requires playing several tricks known from the 3D gaming area to achieve good optical results.

Up to now the increasing performance of graphic accelerators was partly compensated by higher resolution displays used in automotive. Soon the resolution of TFT displays is reaching or even exceeding the resolution of the human eye which might stop this trend unless techniques like 3D displays push the requirements even further.

Taking into account the further increasing performance of graphic accelerators rendering photo realistic images will become more and more feasible. The perfect outcome of such a rendered mechanical instrument cluster however looks like what was put in the car years ago.

In the future significant effort will be put into the development of technologies to make use of the new capabilities for improved HMI ergonomics. This is a big field that can be actively used as a differentiating factor. The general idea behind potential developments is to present more complex information in a simple way.

Examples of such evolutions we already see in cars today are e.g. the economy display visualizing the driving style by the amount of green leaves on the display. There are many potential improvements foreseeable in that area like:

- **Skins**: capability for the user to define the look and feel by providing individual skins.
- **Advanced HMI design**, e.g.
  - Focusing the attention of the driver by using blurring
  - Active use of colors to attract the driver's attention. E.g. change color of the rendered speed needle when exceeding the speed limit.
- **Context aware displays**: Dynamic partitioning of the display – depending on the situation flipping away gauges and show more relevant information for the given situation.
- **Driver distraction management**: Adapting the information presented to the driver to the actual driving conditions to prevent accidents.
- **Augmented reality**: especially head-up displays are well suited to actively overlay the reality with graphical elements in order to provide additional information to the driver or highlight critical situations like crossing pedestrians.

The consumer world demonstrated in recent years that the HMI is an important component for customers and revolutionary new concepts significantly changed the market. Although it is not foreseeable what will become standard in future cars it is obvious that the instrument cluster has the potential to make a real difference here. As it is the main source of information for the driver it
has significant impact on the overall perception of a car.

6 References
[0] Daniel Rice, Robert J. Simpson: OpenVG 1.1 Specification. www.khronos.org/opengv


About the author
Michael Staudenmaier received the diploma degree in electrical engineering from the Munich University of Technology in 1994. Since 1999 he is working as Systems Engineer for Freescale Halbleiter where he is responsible for definition and development of multimedia IP in the application area of consumer electronics and driver information systems. His focus area is the use of computer graphic technology in the automotive embedded space.

This article originally appeared on EE Times Europe.

Related links:
Automotive electronics: What's hot in 2012
Maximizing OpenGL(R) ES 2.0 for Embedded Applications
3D Graphics for In-Car Navigation and Infotainment Systems
Voice recognition cuts driver distraction
Why HTML5 Is Becoming the HMI Technology of Choice
Beyond the code: How designers and programmers can work together to deliver successful GUIs for embedded systems