Mixed Resolution Graphics Technology

Makoto Ono, IBM Corp. (onom@us.ibm.com) Paul Puey, Jeff Bolz, NVIDIA Corp. (paul, jbolz@nvidia.com)

CR Categories: I.3.4 [Computer Graphics]: Graphics Utilities – Software support, Device drivers; I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction

Keywords: high resolution display, usability, mixed resolution

1 High Resolution – benefits and problems

Due to the LCD display technology revolution, the latest pixel pitches are over 200 ppi (pixels per inch), which is over twice the density of the very common 15" 1024x768 pixel displays (85 ppi). Higher density displays can help users understand images (such as satellite images or medical images), and can support more realistic rendering

On the other hand, if the software is not aware of the pixel density (and physical pixel size), the rendered object becomes physically smaller on a higher density display. Though most OSs provide methods to obtain pixel density, many programs tend to define the key user interface components using pixel counts that make GUI objects too small to use on higher resolution displays.

2 Mixing Resolution on a Single Display

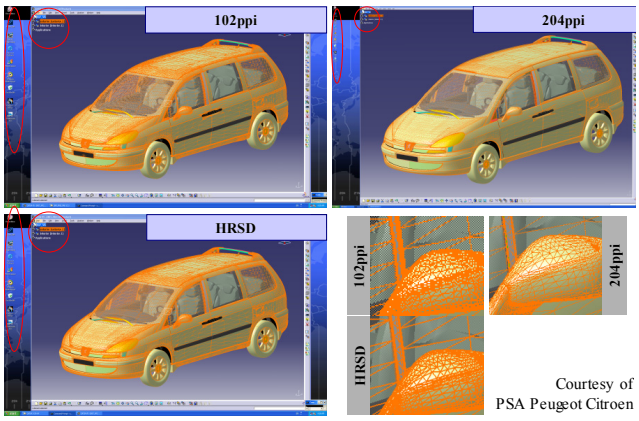

Figure 1. Mixing Resolution on a Single Display

There are several approaches to this issue. Areal zooming utilities[1] or changing the display resolution with a hot key are typical, but popup menus don't work well. Another approach tried to directly manage the OS built-in user interface components[2]. Unfortunately this still has problems, for example, when the software application creates the user interface components by itself.

The goal of our work is to render images or shaded models using the higher pitch (more pixels per inch), but to draw user interface components that should be large in traditional ppi. In order to support this approach, there are two key design points:

(1) How to detect the resolution (ppi) used to render the objects.
(2) How to fill in the gaps between the two coordinate systems.

Furuichi et al.[3] invented a method to realize a mixed resolution environment. However they have not showed an actual implementation of 3D graphics combined with GUI window systems.

3 Application Transparency

Our work is to support mixing the resolution through (1) defining a 3D API (such as an OpenGL API) results that should be rendered at the higher resolution and leaving the 2D API results at lower resolutions.

The benefit of this approach is **application transparency**. If the software uses the OpenGL API to render images or polygons that should be rendered at a fine pitch, and uses the 2D API to draw the user interface components, everything works automatically. The key question is Question (2), the coordinate system gap. A Window system sends the viewport size at the lower resolution to the software and the software renders objects (using the OpenGL) under the lower resolution coordinate system (i.e. the Current Raster Position would be defined using the lower resolution coordinate system).

Our solution is that (1) the OpenGL coordinate system works at a lower resolution, (2) an OpenGL pixel has space to hold subpixels, and (3) a fragment generator creates additional pixel information for fine pitch rendering and stores it in the subpixel space. The CRTC will scan all of the pixel data and use a higher resolution refresh rate, expanding 2D pixels and aligning OpenGL subpixel information accordingly.

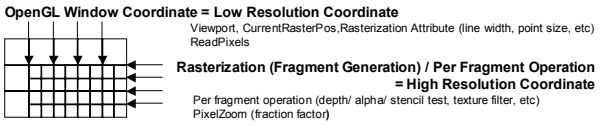

Figure 2. HRSD Coordinate Systems

4 Conclusions

HRSD (High Resolution Scalable Desktop) developed at NVIDIA implements our work and we've tested it with application software. Most programs passed, especially when the coordinate system was defined using floating point. Some pixel-based imaging software didn't work since glReadPixels returned the averaged value for low resolution coordinates. While this could interfere with our approach, we bridged this gap by providing an OpenGL extension to define the glReadPixels target resolution.

References

[1] NVIDIA FORCEWARE (http://www.nvidia.com/page/software.html)

[2] PORTRAIT DISPLAYS LIQUIDVIEW®, LIQUIDSURF® (http://www.portrait.com/)

[3] FURUICHI, S., OHARA, M., AND KAWASE, K., 2002. *Published Patent Application in Japan # 2002-27852*