Acceleration mechanisms in graphics frameworks for User Interface design

Jithu Niruthambath and Prabindh Sundareson, Texas Instruments

User Interfaces (UIs) have transformed from plainly encapsulating the functionality of a device to capturing the intent of user operations and optimizing the overall control flow. Depending on the end application, the appearance of UIs can range from simple clickable rectangles to photo-realistic, physically modeled objects. UIs also need to cater to different resolutions from small liquid crystal display (LCD) screens to external high-definition (HD) monitors.

Gauging the performance of an User Interface
Performances of UIs are measured by how fast the framework responds to user commands and completion speed of the operation. Typical response time requirements are in the sub-100 millisecond range when text and other 2D elements are on a screen.

Given these requirements, it is important for framework developers to separate the visual aspects from the backend rendering elements of the framework. Proprietary software frameworks have given way to UIs built on open and open-source frameworks such as Android, Qt, Flash, HTML5, among others [1,2]. These frameworks abstract the visual presentation from the functionality of the framework and low-level rendering.

Semiconductor vendors such as Texas Instruments (TI) offer high performance processors with ARM Cortex-A cores and advanced graphics capabilities to accelerate UI performance.

Acceleration “fast-paths” are provided by two mechanisms in current frameworks:

1. Utilizing a NEON coprocessor for acceleration of floating point-operations (involved in blending, resizing and rotation operations),
2. Using OpenGL ES 2.0 API for 2D and 3D acceleration.

The NEON coprocessor from ARM is useful in cases where there is enough ARM performance available to support rendering operations. The Pixman application programming interface (API) [3], for example, provides a low-level API that is easy to integrate.

OpenGL ES 2.0 is a more complex-state, machine-based API. OpenGL ES 2.0 offers flexibility to create special effects in 3D, as well as accelerate standard 2D raster operations. Standard 2D operations are mapped through texturing operations, clipping and blending, but operations like live texture updates are inefficient in the standard frameworks using APIs like glTexImage2D. These use-cases are encountered in applications such as live update of camera feeds in video surveillance

These special use-cases are handled with OpenGL ES extensions that can be queried for availability by the UI framework. If the extension is not available, a “slow-path” backup is taken. In order to facilitate easy usage of these extensions, processor vendors like TI provide complete example code, along with demonstrations that showcase these capabilities in Android, Qt and other frameworks.

Summary
As UIs requirements grow to be more complex, it is necessary for software developers to understand the fast and slow paths within standard frameworks based on the underlying capabilities of the processor. Equipped with a better understanding of software, tools, acceleration mechanism and specific extensions, developers are empowered to evaluate and optimize performance of UIs.

References
1. Qt for TI platforms - http://processors.wiki.ti.com/index.php/Building_Qt [1]
2. Android for TI platforms - http://code.google.com/p/rowboat/ [2]
3. Pixman API - http://cgit.freedesktop.org/pixman/ [3]
4. GLES Extensions for image streaming – http://processors.wiki.ti.com/index.php/OpenGLES_Texture_Streaming_-_bc-cat_User_Guide [4]

Source URL (retrieved on *12/25/2014 - 6:56pm*):
http://www.ecnmag.com/articles/2011/02/acceleration-mechanisms-graphics-frameworks-user-interface-design?qt-most_popular=0

Links:
[1] http://processors.wiki.ti.com/index.php/Building_Qt
[2] http://code.google.com/p/rowboat/

[3] http://cgit.freedesktop.org/pixman/
[4] http://processors.wiki.ti.com/index.php/OpenGLES_Texture_Streaming_-_bc-cat_User_Guide