2D Rendering on MIPS® Android

Document Number: MD00809
Revision 01.00
February 3, 2011

This application note refers to Android 2.2 "Froyo" code-base.

Android offers couple of points where 2D graphic rendering can be accelerated using platform specific hardware acceleration provided by destination platform. These points are present in:

- OpenGL module
- Skia module

Android relies on OpenGL ES standard for 3D graphic operations and some basic 2D graphic operations (like blitting two surfaces). This standard is wrapped with two libraries: `libagl` and `libhgl`. These libraries have the same API but implement it in different way. `libagl` is pure software implementation that relies on the CPU (MIPS) for actual processing. `libhgl` is hardware accelerated library that uses GA (graphic accelerator) for acceleration of the 2D/3D graphic operations. Depending on what is present in the system, Android will use either `libagl` or `libhgl`. MIPS Android source tree doesn't have `libhgl` source code, as this library is platform specific and relies on the fact that platform must include GA that supports OpenGL ES standard. `libagl`, on the other hand is implemented and includes MIPS code generation for API functions. In order to truly accelerate 3D (and some basic 2D) rendering, `libhgl` implementation is mandatory. As API for this library is not available, usually GA manufacturers provide their own `libhgl` implementation.

There is one way to accelerate blitting operations (for whole surfaces, especially used during surface composition process in SurfaceFlinger) without implementing `libhgl`. This is possible by implementing library named: `libcopybit`. This library (if present in system) will substitute blitting and stretch blitting operations for surfaces with platform specific implementations (like DirectFB, etc). Library only implements two functions:

- `blit_copybit`
- `stretch_copybit`

Beside `libcopybit` library Android provides another hardware-specific library for managing graphic operations. This library is called: `gralloc`. `gralloc` (short from graphic allocation) library is needed when `/dev/fb*` (or `/dev/graphics/fb*`) device is not available in the system (for any reasons). This is very important, as Android determines/sets screen information and allocates surfaces using `graphics/fb*` commands for that device. In order to avoid this (or if your platform doesn't support this device) you will have to put platform specific implementation of the graphic allocation operations in this library.

The Skia module is primarily used for resource rendering, like icons, images, etc. This module also supports introduction of the platform specific acceleration, but not on the library level, rather on the function level. These functions include:

- `S32_D565_Opaque`
- `S32_D565_Blend`
- `S32A_D565_Opaque`

- `S32A_D565_Blend`
- `S32_D565_Opaque_Dither`
- `S32_D565_Blend_Dither`
- `S32A_D565_Opaque_Dither`
- `S32A_D565_Blend_Dither`
- `S32_D4444_Opaque`
- `S32_D4444_Blend`
- `S32A_D4444_Opaque`
- `S32A_D4444_Blend`
- `S32_D4444_Opaque_Dither`
- `S32_D4444_Blend_Dither`
- `S32A_D4444_Opaque_Dither`
- `S32A_D4444_Blend_Dither`
- `S32_Opaque`
- `S32_Blend`
- `S32A_Opaque`
- `S32A_Blend`

These functions perform several different blitting operations (with dither, blend, etc) on the row level for several different color formats. In MIPS Android source tree some of these functions are implemented using MIPS DSP ASE instruction set (both rev 1 and 2). Beside blitting, Skia performs also scaling, rotation and perspective operations which are present in Matrix module. These functions include:

- `ClampX_ClampY_nofilter_scale`
- `ClampX_ClampY _filter_scale`
- `ClampX_ClampY_nofilter_affine`
- `ClampX_ClampY_filter_affine`
- `ClampX_ClampY_nofilter_persp`
- `ClampX_ClampY_filter_persp`
- `RepeatX_RepeatY_nofilter_scale`

- `RepeatX_RepeatY_filter_scale`
- `RepeatX_RepeatY_nofilter_affine`
- `RepeatX_RepeatY_filter_affine`
- `RepeatX_RepeatY_nofilter_persp`
- `RepeatX_RepeatY_filter_persp`
- `Filter_32_alpha`
- `Filter_32_opaque`

In MIPS Android source tree these functions are also implemented using MIPS DSP ASE instruction set.

All of previously mentioned Skia routines are on row/pixel level. This approach is not suited for optimization of the Skia library using other hardware accelerated graphic libraries (like DirectFB), as the overhead of inserting new library calls will exceed performance gain for small amount of data. In order to fully utilize acceleration of the other graphic libraries, implementation will have to be on frame/surface/rectangular level, which is complex task.

Document Revision History

Revision	Date	Description
01.00	February 3, 2011	Initial version.