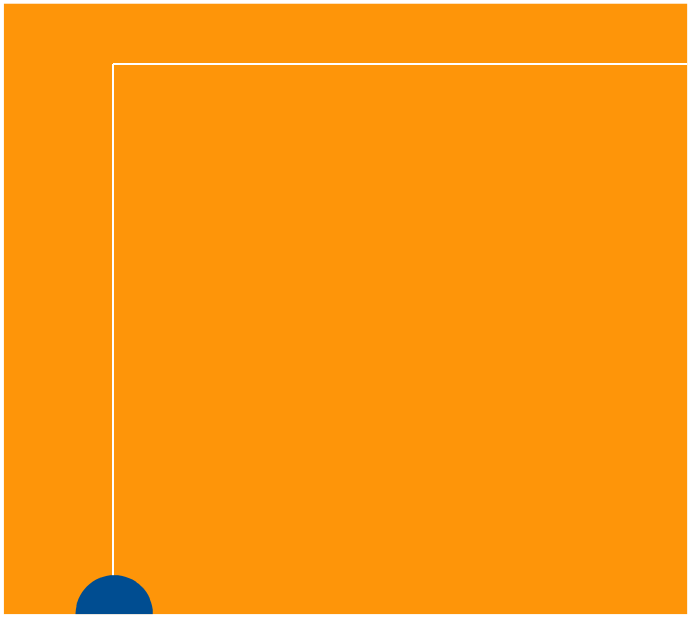
MOTOROLA
intelligence everywhere™

Performance improvement tips in M3G games

March 1, 2006



WHITE PAPER

Performance improvement tips in M3G games

By

Motocoder Staff

JSR184 M3G Mobile 3D Graphics™ is a concise set of 3D API interfaces for mobile 3D applications, which is supported by Motorola E680/E680i/A780. M3G is only a set of high-level java interfaces; the low-level implementation in E680/E680I/A780 is the Swerve 3D engine made by SuperScape which is a high performance 3D engine for Arm/XScale processor.

Below are some technical tips for the performance improvement of J2ME 3D games. Remember, what game players really care about is what the game looks like, not how it is implemented.

Basic concepts in M3G

In M3G, the Graphics3D renders the interface, the World is the whole 3D scene, and the Camera is the view point. 3D objects are organized by a tree structure and defined by points, polygons and meshes. Transformations such as move, rotate and scale are implemented by matrix multiplex.

Technical Tips

Limit the total polygon amount

For Motorola E680/E680i/A780, it is highly suggested that the total amount of polygons of a 3D scene (including characters, objects and environment) must not exceed 1500.

Mix 2D and 3D objects

In J2ME 3D games, the 2D object consumes far fewer resources than the 3D object, but sometimes the effect looks similar. So it's a good idea to replace some 3D objects with 2D slides in the games. For example, clouds, snowflakes or billboards can be easily converted to 2d images. Sometimes, drawing directly in the background image will also reduce resource consumption.

Simulate animation with texture

In 3D games, there are usually small objects needing relatively complex animation effects to make them display realistically. For example, to represent cars running in a small-screen game, you'll need some animation effect to let the player know that the car is actually running. But it is unpractical and unnecessary to use all real 3D models to simulate this. An alternative is to change the texture in the object, to make it *look like* it is running.

Divide large mesh into several parts

The swerve engine used in the E680/E680I/A780 supports geometry culling (or meshes culling). A cube in a view port, for example, has invisible faces that will be ignored. But a mesh will only be ignored if none of it is visible.

That is to say, if you have a mesh that covers a huge area (e.g. road) and is currently exported as a single mesh object, the performance will be better if the mesh is chopped up into a few sub-meshes, because only the sub-mesh in the view port needs to be calculated and rendered.

But a balance needs to be found between splitting the object into multiple parts. The more parts, the more processing is required as the scene tree is larger. With fewer parts, large objects that are only partly visible can not be culled. Finding the balance is going to require experimentation.

Replace the math functions with hash tables

E680/E680I/A780 use Xscale 300MHz as their processor. In fact, like most CPUs in mobile phones, the CPUs in E680/E680I/A780 do not have a float point co-processor in it – that is to say the CPU has to simulate the float point processing with integer processing, which will take a lot of time.

The functions like *divide*, *sin*, *cos* and *square root* will take a lot of computation resources. In the mobile 3D game, the requirement for function precision is low. Using hash tables (function variable as key and function result as value) instead of real functions will upgrade the float point performance.

Key	Value
Sin(0)	0
Sin(5π/180)	0.087155743
Sin(10π/180)	0.173648178
Sin(15π/180)	0.258819045
…	…

Table 1: hash table use to simulate *sin*

Use self illuminated texture instead of light source

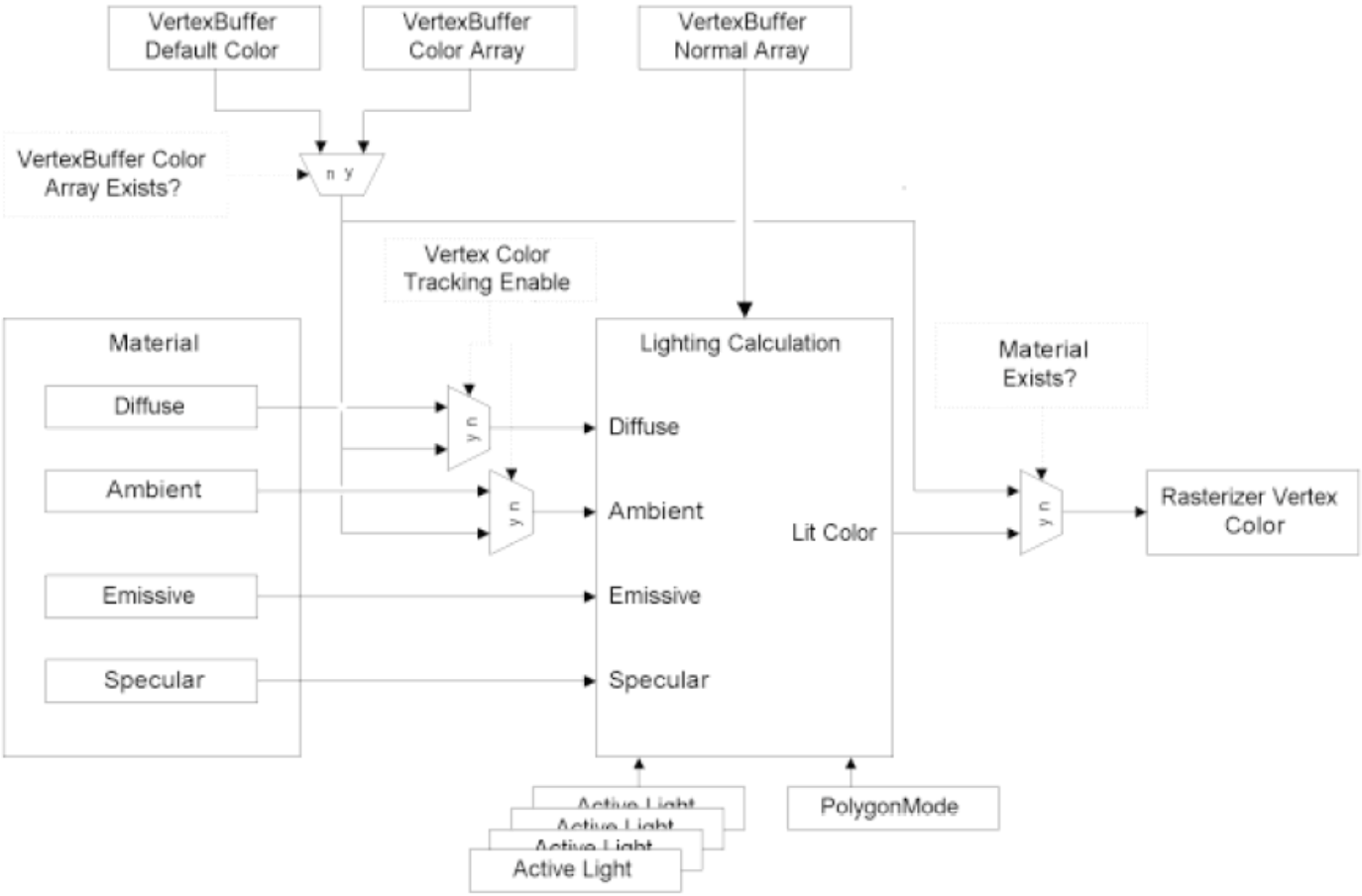


Figure 1: Material and vertex coloring pipeline – figure from JSR184 document

As the figure shows, light calculation is over material. Lights – even directional lights – use a lot of resources. Think instead of creating your textures as if their objects were illuminated.

Note

In mobile phones, the memory size and computation resources are limited, so the point of a good J2ME 3D game is to balance the view, effects and performance. Experiments in real handsets to get the balance are always necessary.

References

JSR184: http://jcp.org/en/jsr/detail?id=184
Digital Red Company: http://www.worldup.com/
Swerve: http://www.superscape.com/