Evaluation of Java 3D
for mobile platforms with M3G

Fredrik Nygren
Jonas Nilsson

September 2004

Lately Java has almost become a standard on mobile phones. This gives developers the opportunity to create applications that are no longer restricted to a limited set of devices without recompilation and modification of its source code. But the mobile phone market is never resting and one of the latest requirements for mobile phones with Java is 3D support.

For J2SE the optional package Java 3D exists and brings 3D support. But since mobile platforms have much less processing power, memory and less hardware support a much more limited option was needed for J2ME (Java 2 Micro Edition). To meet these demands an expert group from The Java Community Process (JCP) formed the JSR-184, or M3G, specification for how 3D could be handled by Java in mobile phones. It was finally release on the 22nd of December 2003 and in the master's thesis *Evaluation of Java 3D for mobile platforms with M3G* [1] the primary objective is to investigate the level of 3D complexity that is possible in games and other applications on a state of the art mobile phone using this M3G specification of Java 3D. The evaluation is made by comparing the usage of Java 3D as it is implemented in the optional package for J2SE with the fragment sized M3G for the J2ME platform.

Since M3G only is an API for 3D graphics it is only this part of Java 3D that is comparable. Java 3D also supports sound, user-input, triggers and basic collision detection. Both the APIs use scene graphs to represent objects in the 3D world and both support immediate and retained rendering modes.

To find weaknesses in the M3G API in this master's thesis a game for the Java 3D/J2SE platform was first developed which had an as general game engine as possible. Then a game for the mobile platform was developed based on the same engine structure to find advantages and disadvantages in M3G. During the creation of the M3G game the lack of some convenient functionality such as vector and quaternion mathematics was noticeable. M3G would also have benefited from better support for billboards - flat objects whose surface is somehow aligned with the viewer, commonly used for displaying simple trees. Further on, the way fog was handled by the two APIs differed completely. In Java 3D fog behaved like expected but in M3G the thickness of the fog depended on the direction the camera was looking, this was likely caused by the intention to speed rendering up as much as possible. A feature that was not tested was the file format defined by M3G. It has its own, while Java 3D relies on existing formats which caused problems since these formats were outdated. The reason the M3G file format was not tested is that no export tool existed at the time needed.

Because of the limited time a master's thesis offers none of the two games were made perfect, but their purpose to work as a benchmark to show deficiencies in the minimal M3G API was fulfilled. With the result in hand M3G stood up to the Java 3D very well, being less complex was not always a disadvantage.

---

[1] F. Nygren and J. Nilsson, *Evaluation of Java for mobile platforms with M3G*, Lunds tekniska högskola, 2004.