Run–Time Code Generation for Materials

Stephan Reiter*

Typical material systems are based on a collection of properties that are set on a per–object or per–surface basis usually by evaluation of a description file at run–time. Shading of hit points is then implemented by interpreting these properties. In order to be able to support a large feature set, different code paths need to be created, e.g. for reflection and refraction effects. Conditions for the branches associated with these fragments of functionality need to be evaluated for each shaded ray, which may degrade performance considerably. Material specific optimization opportunities are also missed out by having a generic function for all material configurations.

We propose the use of run–time code generation for materials. This retains the flexibility of a run–time parameterizable material system and allows for performance improvement by elimination of branches and folding of constants. Noting that code generation is non–trivial and time consuming to implement, particularly when dealing with multiple platforms, we suggest the use of the low–level virtual machine (LLVM) [1] for this purpose.

The LLVM is an abstraction from hardware and based on a virtual instruction set. It supports emission of code through an object–oriented interface and run–time generation of machine code for a large number of platforms. Interoperation with C/C++ code is possible through dynamic linking, which is handled transparently by the LLVM. A large number of optimization passes are available that can be applied selectively to balance compilation times and the quality of the generated code.

We present an implementation of the extensive material system from id Software's id Tech 3 engine, which supports multiple texture layers, blending and texture coordinate modifiers among other features. This implementation is used in a new rendering backend for said engine, which allows playing original games based on the engine with raytraced graphics and new effects. With support from the development community we have been able to test this with id Software's Quake 3 Arena and Raven Software's Star Trek Voyager: Elite Force.

Figure 1: Animated fire in the map *q3dm1* from id Software's Quake 3 Arena, rendered at a resolution of 800x600 pixels at 5 fps on an AMD Athlon X2 3800+.

Figure 2: Raytraced reflections on a console in Raven Software's Star Trek Voyager: Elite Force, rendered at a resolution of 800x600 pixels at 2 fps on an AMD Athlon X2 3800+.

References

[1] C. Lattner. LLVM: An Infrastructure for Multi-Stage Optimization. Master's thesis, Computer Science Dept., University of Illinois at Urbana-Champaign, Urbana, IL, Dec 2002.

*e-mail: stephan.reiter@gmail.com