

High quality previewing of shading and lighting for Killzone3

Francesco Giordana
Guerrilla Games - SCEE



Figure 1: Screen captures from Maya for KILLZONE3™ © Guerrilla Games - Sony CEE

1. Introduction

Current generation games are pushing real-time computer graphics to the limits of their potential, getting closer to cinematic quality everyday. With the increase in quality, it has become crucial to preview the visual appearance of the assets with the highest fidelity possible. For Killzone3, after realizing that existing software couldn't reach anywhere near the quality that we were looking for, we decided to try something new. We embedded our game rendering engine inside Autodesk Maya, our main content creation tool, achieving interactive previews with an unprecedented mix of flexibility and quality.

2. The Framework

We developed a Maya-centered framework, where our assets are rendered using our custom OpenGL rendering engine. There are two rendering modes: a direct one, based on custom hardware shader nodes in a standard Maya viewport; and a deferred one, based on a custom deferred renderer that completely overrides Maya's rendering pipeline. The shading is done through CG shaders deduced from a live traversal of the Maya's shading graph, granting us great flexibility in the definition of the materials.

2.1 Hardware shader nodes

Through the Maya API, we created a custom hardware shader node, which, connected to any shading graph inside Maya, will override its rendering operations in the viewport. Each change in the network will trigger a traversal of the graph, through which we build the CG source code for a real time shader to be used for the preview. The resulting shader is then used in combination with the vertex and index arrays provided by Maya to render the geometry using standard OpenGL calls. Up to 8 lights can be used at the same time for this rendering mode. Additionally, we implemented custom pre and post-render operations to perform scene analysis that can greatly increase performance.

2.2 Integrated deferred rendering engine

Truly unique to our framework is our custom renderer, based on our game's deferred rendering engine, which takes over the entire rendering pipeline inside the Maya viewport. This engine has its own scene graph and representation for all objects in the scene. Being a deferred renderer, we can preview any number of lights, with their respective real-time shadows, and also apply a full range of post-processing effects. In general, anything that can be rendered inside the game can be previewed inside Maya, with a quality almost identical to the final in-game result. Thanks to some recent improvements in the Maya API, we are now able to render on top of the scene any elements from the standard Maya rendering pipeline, like manipulators, light representations, wireframes, and similar. The rendering engine also features a series of very useful debugging options, like the visualization of the individual buffers used for the deferred rendering.

2.3 Keeping it in sync

During the Maya session, we keep track of all objects created and deleted by the user via a complex system of callbacks. We build a second representation of the scene in memory, readable by our engine, and we update it every time there is a user interaction. The main objects that need to be synced are meshes and lights: the mesh geometry comes directly from the index and vertex arrays provided by Maya, while the lights get converted into CG shaders similarly to the geometry shaders.

3. Conclusions

The development of this framework has presented many challenges, given that we pushed Maya into a brand new territory, but the result has proven to be a very successful improvement to our pipeline. The artists are now capable of working on any asset with a very high quality preview; this has considerably cut down iteration times for all art teams, becoming an essential tool especially for lighters and shader artists. Future work will be mainly focused on switching to a 64-bit architecture to allow previewing of bigger scenes.