A System For Crowd Rendering

John Haddon
The Moving Picture Company

Dave Griffiths
The Moving Picture Company

1 Introduction

For the films Troy and Kingdom of Heaven, The Moving Picture company was tasked with the creation of hordes of battling soldiers numbering in the order of 100,000 individuals. This sketch describes the internal tools developed to achieve this task within RenderMan, and their evolution for use on more recent projects and with other renderers.

2 Geometry generation

The output from MPC’s custom crowd simulation software consists of a cache file containing a uniquely blended pose for each agent in the simulation. This made it unfeasible to address the rendering with a RIB archive instancing solution [Hery 2001]. Instead we undertook to individually skin each agent to the simulated skeleton on the fly within a procedural DSO.

Rather than write a DSO focused on one particular task, we instead created a C++ library of useful classes and geometry processing routines, and bound this into the scripting language Lua [Jerusalimscy et al 1996]. A small DSO then ran Lua scripts at rendertime, with the scripts in complete control of what was rendered. This allowed us to quickly develop different scripts to adapt to the changing requirements of the project.

A typical Lua script for an agent would read a description format containing a list of objects making up the character, loading each object in turn, skinning or parenting it onto the skeleton for an agent and then rendering it. A master script controlled this process for the entire crowd by arranging all the agents in a spatial tree structure, and recursively traversing it to emit agents only when they were needed, and only at the level of detail required. This process used RiProcedural calls to minimise memory usage, making even the most complex shots manageable.

3 Preview rendering

During simulation, only a relatively crude view of the agents was available. For simulation approval we required a fast but reasonably accurate way of portraying final skinned characters. We achieved this by writing a RenderMan compatible OpenGL based preview renderer, for performing the equivalent of playblasts. By combining this with a script editor we were also able to create a simple interactive development environment, with immediate visual feedback on the results of the scripts being developed.

4 Shading

One relatively simple but flexible shader was used for all surfaces within the crowds, allowing us to take advantage of PRMan’s combined shading for much accelerated rendering. Shader parameters were set procedurally from within the Lua scripts, defined according to variation ranges set by the lighting artist. We then implemented a simple image based selection tool to allow the artist to select any agent or group of agents and override the procedural choice of shading parameters with a more explicit choice. This tool could not only modify shading but also agent type, clothing and weaponry, although its most common use was to remove unwanted agents entirely.

We also developed a shadeop to provide a rapid approximation of ambient occlusion without the time or memory overheads of raytracing. Occlusion from each agent was approximated by a collection of ellipsoid fields to represent the major body parts, with upwards of 1,000,000 fields used to represent an entire crowd. These fields have a limited realm of influence, and by accelerating their lookup with a tree data structure we were able to render film resolution frames in very reasonable times.

5 The aftermath

Following on from Kingdom of Heaven, we generalised the system to support Mental Ray in addition to RenderMan, and extended it for general use across all our film shows, rendering aliens, bats, oompah loompahs, furry things and cityscapes in the process. We are also using Lua to implement scripted nodes, deformers and fields within Maya, and using an updated OpenGL renderer to visualise script results directly within the Maya viewport. From this we have concluded that a combination of a C++ library and a lightweight scripting language provide not only efficient means of rendering complex scenes, but also a valuable tool in a wider film pipeline.