Dan Knowlton CG Software Engineer

dtknowlton@gmail.com | (540) 846-8378 | www.danknowlton.com

Demo Reel Breakdown

1) Fluid Simulation

- MAC-Grid based Particle-in-Cell fluid simulator with support for OBJ boundary constraints.
- Implemented in C++ and based on the paper "Multiple Interacting Fluids" from SIGGRAPH 2006 by Losasso et al. and the SIGGRAPH Fluids Course Notes by Robert Bridson. Extended fluid simulation base code by Christopher Batty.
- Responsible for viscosity implementation, signed distance field boundaries, level-set and marching cubes integration, and other basic user interaction and scene setup functionality.
- Basic viscosity implemented as a joint project with Yining Karl Li for Physically-Based Animation Final Project.
- Added variable viscosity to the simulation capable of simulating changing material viscosities as well as temperature diffusion/advection.

2) LightWarp: Artistic Volumetric Lighting

- Joint project with Stewart Hills for Advanced Graphics Final Project.
- Implemented a C++ and Maya API plugin for Maya 2012 as well as a custom Renderman shader for rendering curves as "volumes". Maya plugin allows for easy generation and manipulation of curves as well as intuitive controls for the shader.
- Based on the 2011 SIGGRAPH paper "A Programmable System for Artistic Volumetric Lighting" by Nowrouzezahrai et al.

3) Jell-o Simulation

- Implemented in C++ and Maya C++ API for Physically Based Animation.
- Responsible for implementing a mass-spring system with various spring, collision, and contact forces as well as the basic integration scheme.
- Extended the simulator to work within Maya as a deformer node allowing the Jell-o to fully integrate with a Maya scene. I also developed a tutorial for the class on how to integrate the simulation with Maya.

4) Smoke Simulation

- Implemented in C++ and OpenGL for Physically Based Animation.
- Based on SIGGRAPH 2007 Fluid Course Notes by Robert Bridson.
- Simulation based on advecting densities, velocities, and temperatures through a grid with a Semi-Lagrangian approach.

5) Renderer

- Implemented with C++.
- Renderer supports raytrace, contour, depth, normal, and pathtrace renders.
- Includes an OBJ loader and a KD-Tree for optimization.