Creating Realistic CG Honey

Allen Rulova

Dreamworks Animation

Introduction
During the pre-production phase of Bee Movie we faced the challenge of creating realistic looking CG honey. The development approach was divided into two main parts: creating the overall look, and simulating the dynamics. In this sketch I will discuss how we met these challenges by incorporating various elements into a raytrace render, and by applying viscoelasticity in the fluid solver used for the particle simulation.

Look Development
In terms of look, the first choice made was that of hue. Then the refractive, translucent, and reflective properties were explored as well as surface generation technique. Reference footage taken of real honey indicated that we required additional elements including bubbles and caustic patterns. We also attempted to add an opaque haze (not shown in here) using a depth map generated by a raytrace shader. When properly applied this gives the honey a thinner appearance closer to the edges, similar to a fresnel effect.

A blobby surface was created from particles simulated in our fluid solver. A key component to the look we were trying to achieve was to capture folds or inflections in the particle distribution. The image on the far left of Figure 2 shows the mesh having an inflection in the lower right hand side of the surface close to where the geometry meets with the collision plane. Since our blobby surface generation tool utilizes an octree data structure, careful choice of voxel width and input particle radius made capturing these folds possible.

We created the caustic patterns shown in the center image of Figure 2 using a two step pre-render raytrace technique. Particles were fired along rays from an imaginary light, refracted with the front surface of the honey, then intersected with the back side. The particles were mapped back on to the surface at render time, affecting the diffuse and specular components of the shader applied to the honey surface.

The bubble spheres shown in Figure 2 were taken from a subset of the simulation particles. The bubble shader is similar to the one used on the honey surface with slightly different refractive properties. In both cases the surface is highly transparent and specular. All three elements were rendered together in a single pass using a proprietary raytrace renderer. The final results are shown in Figure 1.

Dynamics
We achieved the most realistic motion possible utilizing our proprietary fluid solver. Given the highly viscous nature of real honey, our approach focused on pushing the fluid solver’s viscosity controls as high as possible. Unfortunately this only tended to dampen the motion. It was then clear that we needed to incorporate an additional elastic force to the fluid resulting in the “viscoelastic” motion we required.

In [Bargteil et al., 2004] a technique is described for extending the Navier-Stokes equations to have a term representing elastic strain, which is integrated over the fluid solver's grid. The strain used is actually comprised of two parts, a plastic as well as purely elastic:

\[ \varepsilon_{\text{Tot}} = \varepsilon_{\text{Pl}} + \varepsilon_{\text{Elc}} \]  

(1)

where \( \varepsilon \) is the elastic strain tensor. The proper balance of these two forces ultimately resulted in the behavior we were looking for. Simulation parameters controlled the contributions of each. The elastic component behaved in a rubbery or “jello” like manner while the plastic component contributed when the elastic force exceeded a certain threshold value. It was at this threshold the flow would lose it's stiffness and instead flow in a smooth and viscous manner.

Conclusion
By exploring a technique relatively new to the computer graphics community, one that gives artists a new method for controlling fluid simulations, we successfully replicated the behavior of honey. Although creating the correct look of the honey in terms of color and transparency proved essential and possibly sufficient to be convincing, achieving the proper motion gave the honey an even greater sense of realism.

Acknowledgments
Mahesh Ramasubramanian, Mike Miller, Adam Bargteil, Joshua Schpok, Steve Wood, Scott Cegielski, Xray Halperin

References