Final_Report.md

Dye Drop Rendering

by **Samaksh (Avi) Goyal (sagoyal), Ian Madden (iamadden)**.

Inspiration

The dye drop is a beautiful creation that is often the subject of many desktop backgrounds, artwork, and more. It is also a culmination of several unique and complex fluid dynamical processes, diffusion and turbulence, so the setup of the modeling problem was already very challenging. The material properties of the dye in water is also important: acrylic paint can be shot in a jet-like fashion to have one effect, while a more diffuse, water-color ink drop has an entirely different effect. Our aim was to render this image effectively using the principle of volumetric photon scattering. This would allow us to see the unique behaviors of the medium in scattering light. We hoped that this would generate a few caustic effects, with some coloring effects as well. Finally, we wanted to set this beautiful imagery in the place we believed such an image would properly belong: a modern art exhibition.

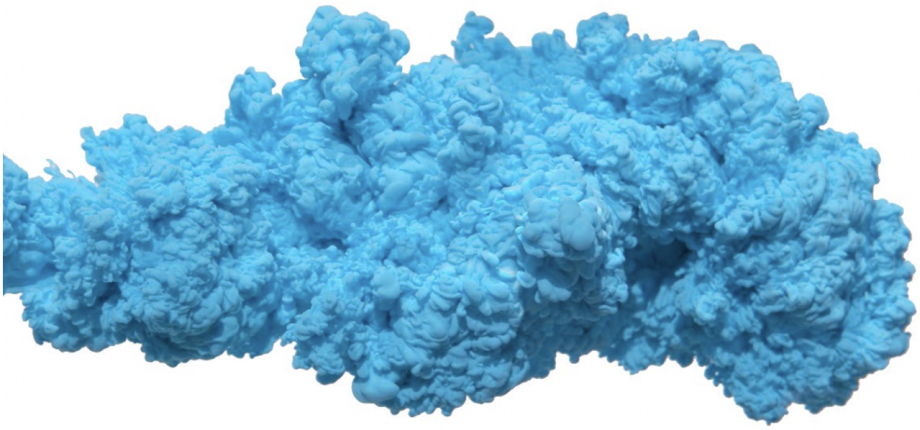
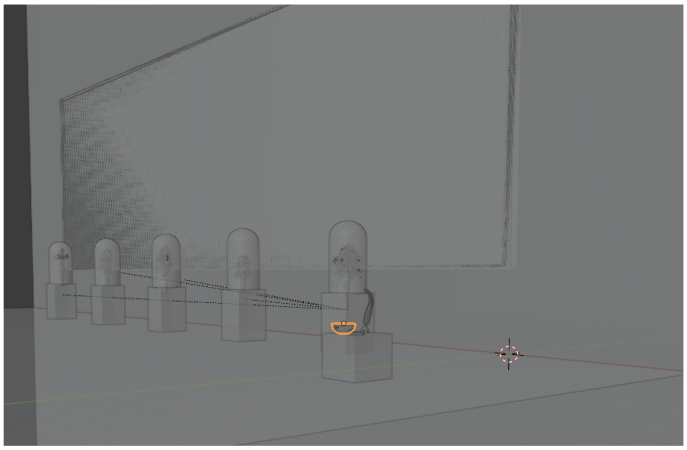


Exhibit Hall

We imagine an exhibit of acrylic paint jets being pushed into a chamber of water in an art museum, while at the near corner, we see a quill dipped in a bottle of ink. We see a drop of dye from the quill diffusing in the glass bowl, and a journal for use of the writing. We did not focus on the minute features of the feather, but rather the volumetric effects of these fluids diffusing in water.

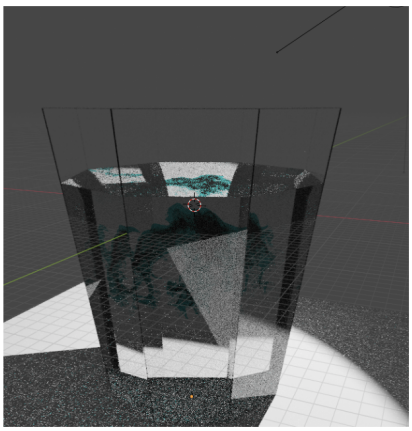


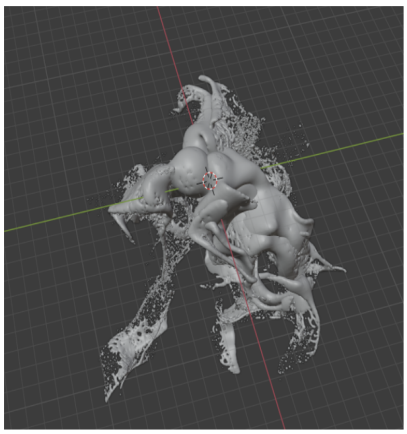
Ink Splash Scene

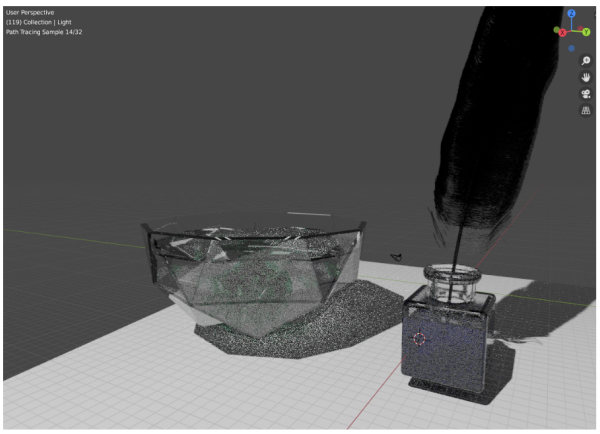
In order to model the splash of ink in the nearby bowl, we discretized the liquid into tiny individual particles, and modeled diffusion through addition of Brownian Motion (acting as a diffusive force), a low gravity (some downward force, but counteracted by a buoyant force), and finally turbulence (to represent the vorticity and the resistance of the water upon the splash. After the simulation with $N = 10^7$ particles, we created a mesh by convolution over the many particles to generate the following model.

We then moved on to creating the other objects of the scene:

- **Bowl.** To model the bowl, we took a convex polyhedron and cut the polyhedron and half, created a shell, and filled with water by using a similar polyhedron.
- **Quill** We modeled a low order Bezier curve, and then used Blender's particle modelling features to grow "hairs". After this, we created a mesh by adding some radius to the hairs.
- **Inkwell** We modeled the inkwell with a cylinder mated to a cube, with some smoothing and remeshing in order to create the appearance.
- **Journal** The model of a journal was borrowed and downloaded (free) courtesy of TurboSquid, "Pen and Journal."







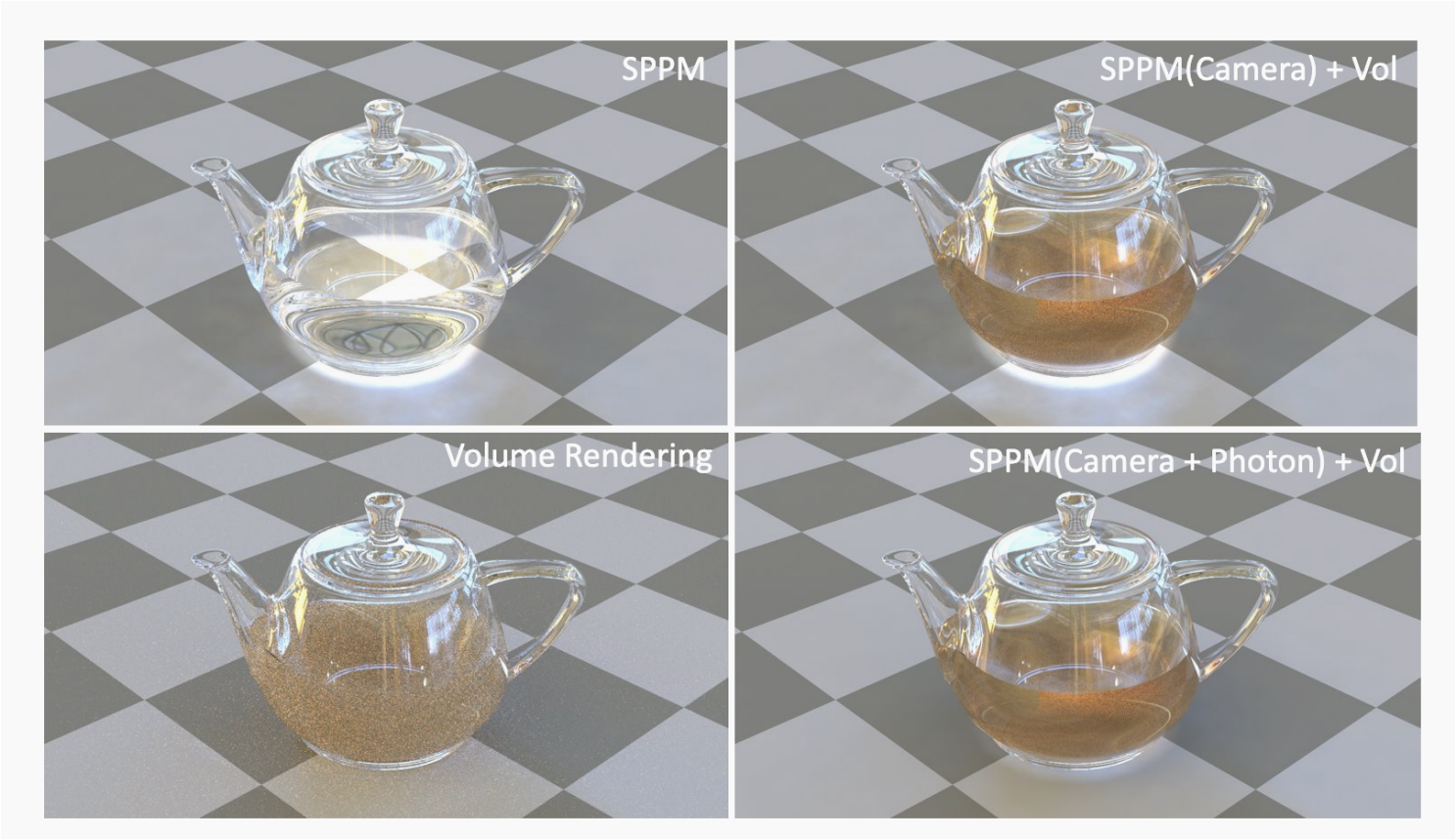
Smoke Vorticity

We developed the large-scale image of the smoke in the scene by running a longer simulation of smoke (with diffusive and Brownian effects, as well as vorticity) [1]. We created a 2D image through this, and developed the painting through placement on a plane as a 2D texture.

Paint Jet Through Water

In order to model the paint jet pushing through the water, we actually combined the knowledge of smoke and ink drop. The model of water was still based on a discretization into particles, and the same turbulence addition still applied. However, the rising behavior was inherited through a "forcing" through the smoke simulation.

Volumetric Photon Mapping Implementation



We implemented volumetric photon mapping to make the realistic lighting effects of light scattering in the glass bowl with a scatting liquid inside. We drew inspiration from significant prior work on the subject [2, 3]. Our implementation concatenates the stochastic progressive photon mapping (SPPM) integrator found in PBRT `sppm.cpp/h` and the volumetric path tracing routine `volpath.cpp/h`.

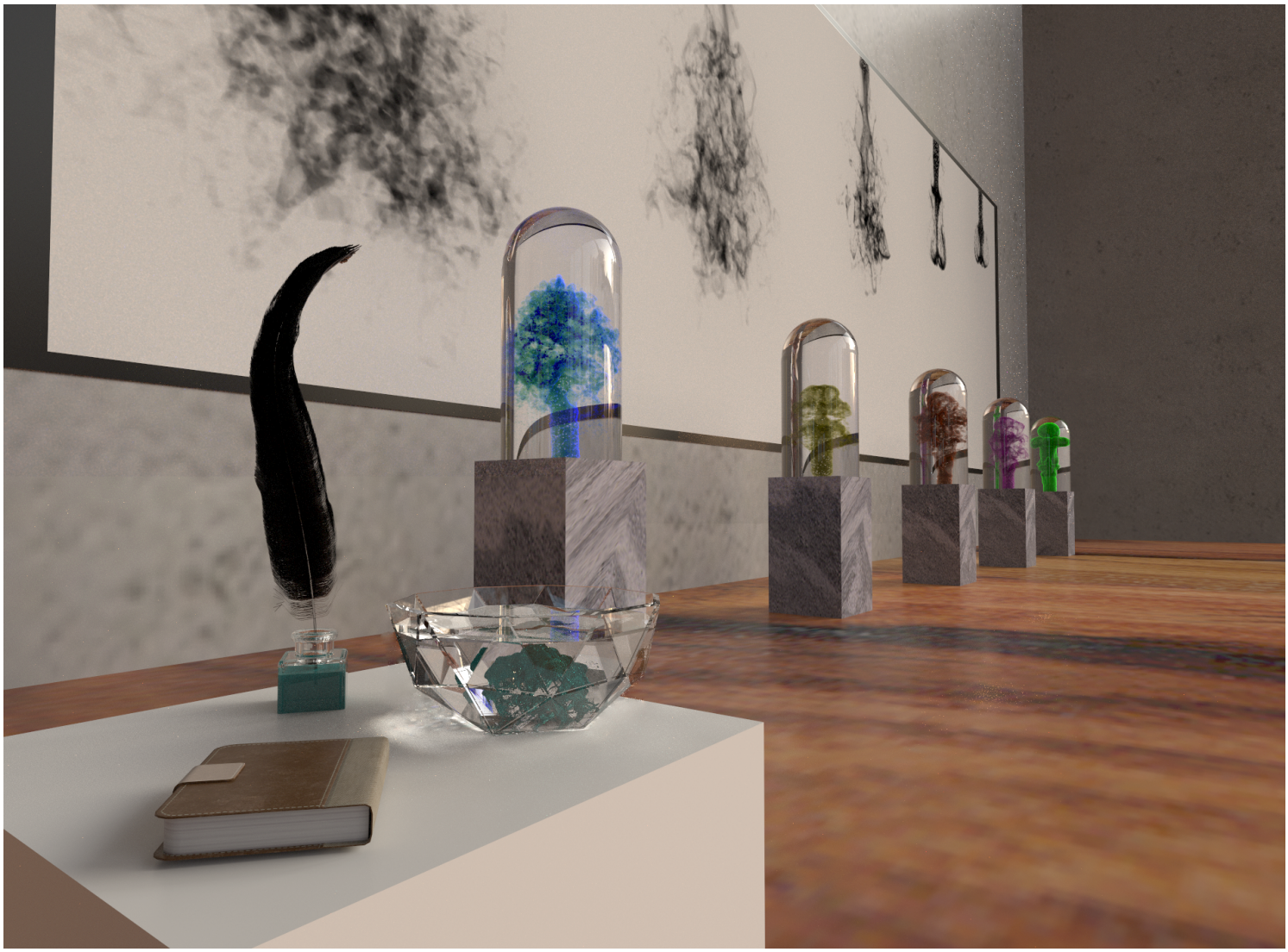
For reference we provide an image of a glass teapot with tea inside with just SPPM integration (top left) and with just Volume scattering (bottom left). The scene pbrt file was graciously made available by Benedikt Bitterli.

Our implementation is in `sppmvol.cpp/h`. We add volume scattering to both phases of the current SPPM implementation. In the first phase of SPPM when the camera ray path deposits visible points on the surface, we insert the main for loop in volpath.cpp to handling scattering at point in medium. This way we can sample in-scattered visible points inside the medium. This addition is visible in the top right image of the figure. The tea has color now because of the scatter effect.

During the second phase of SPPM when we are shooting rays from the light source we need to intersect the photons with the volume medium (so that accumulation can occur). To do so when we hit a volume we calculate the scattering at the point of intersection and then shoot a new photon in the sampled direction. This is visible in the bottom right image of the figure.

We would like to thank the Xianzhe Zhang, Wen Zhou (CS348B class of 2019) for their provided explainations of the above process - this made debugging easier.

Final Render



Render Parameters:

The full resolution is 1900 x 1400, with 2048 halton samples per pixel, 15000 camera iterations and 1000000 photon iterations. and 2048 volumetric sppm integrater numiterations.

Trouble Shooting

We ran into several problems along the way:

- We had trouble rendering our high resolution image in a reasonable amount of time locally so we shifted an AWS EC2 instance with `96` virtual cores.
- We were unable to render images with `10k+` camera ray iterations, so to overcome with Matt suggested we take advantage of the `cropwindow` argument on the `Film` attribute and use `imagtool assemble` to gather cropped images. This idea worked perfectly.
- The radius parameter was especially tricky to set accurately in our `sppmvol.cpp/h` implementation. We found that an initial condition of `radius = 0.2` gave the best results.
- There were many issues with accurate fluid simulation, and porting this over to PBRT. Because there was no clear way to take advantage of the "GridDensityMedium" that was used in other implementations of smoke. As a result, it was necessary to generate a mesh through blender instead of using a density. This necessitated the use of a discretized particle view of liquid and smoke rather than a density view. Material properties, as a result, also had to be done manually as well.

Distribution of Work

We were both equally involved in the scene setup, story-boarding and implementation of rendering strategies. We meet collaboratively on long zoom calls.

Ian Madden: Set up Blender scene, fluid simulation and meshing, material properties, and camera settings, debugged volumetric photon mapping code

Samaksh (Avi) Goyal: Implemented volumetric photon mapping, set up AWS to render image, advised in scene development

Code

Sources

[1] Kim, T., Thürey, N., James, D., & Gross, M. (2008). Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics (TOG)*, *27*(3), 1-6.

[2] Henrik Wann Jensen. 2001. Realistic Image Synthesis Using Photon Mapping. A. K. Peters, Ltd., Natick, MA, USA.

[3] Jensen, Henrik Wann, and Per H. Christensen. "Efficient simulation of light transport in scenes with participating media using photon maps." Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM, 1998

> Written with the help of StackEdit.