Time Travel Effects Pipeline in 'Mr. Peabody & Sherman'

Robert Chen Fangwei Lee David Lipton
DreamWorks Animation*

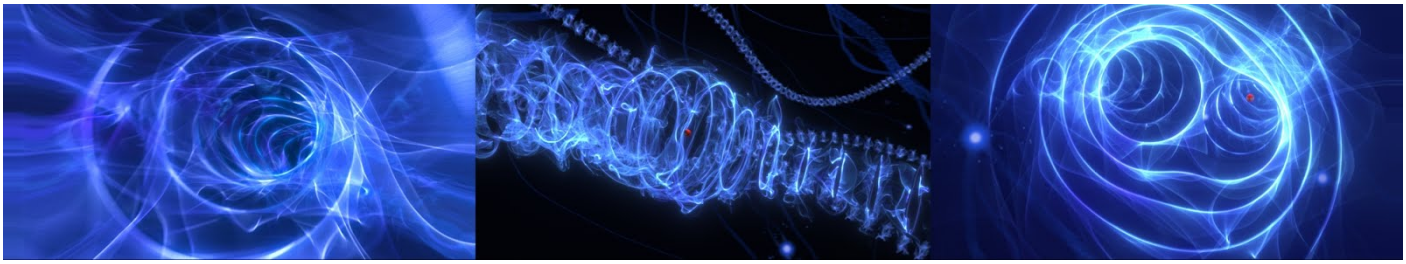


Figure 1. *Various time travel shots from the film: Looking into a single tunnel, looking sideways from outside, and a tunnel that splits into two.*

Abstract

In DreamWorks Animation's *Mr. Peabody & Sherman*, our titular duo travel back in time through magical wormholes (Fig. 2). We needed a system that creates intricate and ethereal wormhole tunnels, which were inspired by the mathematical *fractal flame* images [Draves and Reckase 2008]. Yet, unlike fractal systems, it needs to be highly art-directable and animated so the tunnels can materialize, disintegrate, and branch off as the time machine travels through them. The system also needs to be scalable in order to create lots of tunnels in wide shots while maintaining details in close-up shots. Due to the scale of the effect and its use across multiple shots throughout the show, we've set up a pipeline for the Layout, Animation, Effects, and Lighting departments to collaborate on. The pipeline is divided into several stages, and as artists advance through these stages, the look gets more refined: from the broad strokes to the finest details.

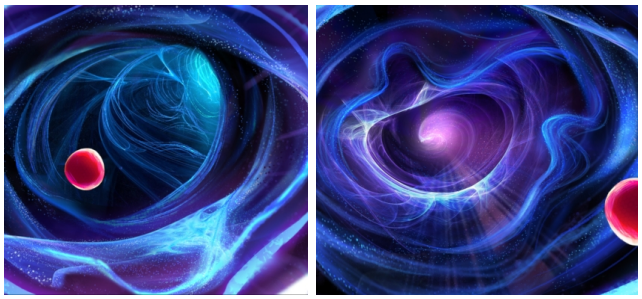


Figure 2. *Concept art*

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - Curve, surface, solid, and object representations; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Fractals;

Keywords: production, animation, effects, fractal flames, simulation, parallelization

© DreamWorks Animation, L.L.C. 2014. This is the author's version of the work. It is posted for your personal use. Not for redistribution. The definitive version was published in Digital Production Symposium 2014, https://doi.org/10.1145.2633374.2633383.

1. Planning the Route

The first stage of the pipeline involves planning out the path of the wormhole tunnel. The Layout department, using a tool that we developed, models out tree-shaped paths and aligns rings along them (Fig. 3). This allows them to quickly preview the wormholes and approve the camera (3.i) early on. The ring geometry is also exported for the Animation department so they can tweak the animation of the time machine.

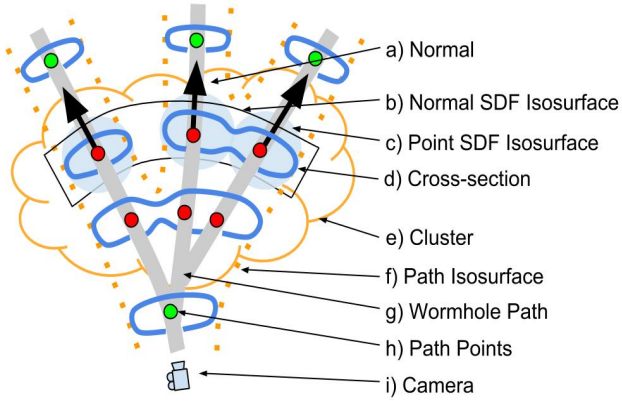


Figure 3. *Cross-section Diagram*

The Effects department then takes these paths from Layout and resamples them into points with equal intervals (3.h) and groups these points into clusters (3.e) by calculating the point cloud density. The clusters represent key tunnel features, such as branching or merging.

We need to create the outlines of the tunnels by converting the point clouds into isosurfaces. Yet, because the paths are very long, it is too expensive to create a single isosurface with enough resolution to mesh the entire tunnel, so we developed a method to approximate the cross-sections of the isosurface. For each individual point or cluster, we generate two isosurfaces: an isosurface from the distance field of the points (3.c) and an isosurface from the distance field of their normal planes (3.b). The latter is approximated by summing the distance field of each normal plane in the cluster with a smooth function. The intersection of the two isosurfaces forms the cross-sections of the path (3.d). We can preview the resulting tunnel outlines (Fig. 4) to make any adjustments before we go into the next stage.

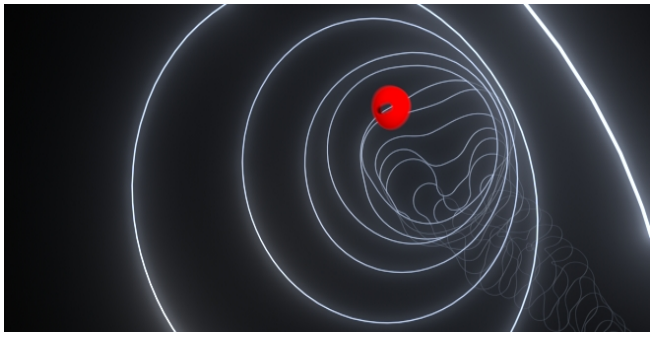


Figure 4. *Wormhole tunnel outlines*

2. Surface Membrane Generation

To create the "skins" of the tunnels with a membranous quality, we use the wormhole outlines to emit multiple overlapping and undulating surfaces. Since the profile rings (Fig. 5.a) are generated from the cross sections of morphing isosurfaces, their topologies are temporally incoherent. To address this, we resample them to a fixed number of points. Using these points as emitters, we emit particles (5.b) that are first advected by curl noise to create the flaring motion, then further advected by the movements of the rings to inherent the morphing motion. The particles are grouped by their emission frames. For each particle in a group, we trace the closest unique neighboring particle in the next frame group and connect them by a line. The process is recursively repeated to create a series of curves with consistent segments (5.c) that are finally lofted into NURBS surfaces (5.d).

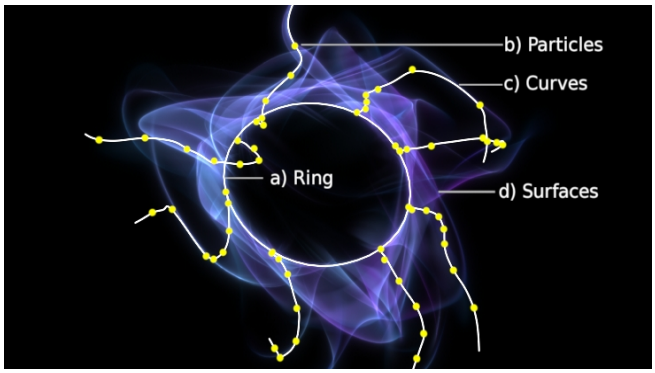


Figure 5. *Surface creation*

On average, a shot consists of hundreds of rings, and each ring emits multiple overlapping surfaces (Fig. 1). It is obvious that the processes have to run in parallel. Yet, the traditional single array distribution for parallel jobs is not adequate for the complexity of the system. Thus, we developed a dynamic 2D-array distribution method where the tasks are first divided by the rings, and then further divided by the surfaces, resulting in each machine only simulating one surface for one ring at a time. When a ring splits into multiple smaller rings, the system dynamically redistributes the overlapping surfaces onto the smaller rings. Finally, at render time, we load all of the data on disk and render them together, resulting in a highly-efficient, massively parallel process with fast turnaround time. Since we only need the surfaces for the amount of time they are visible on screen, we can further optimize the system by recording the frames at which each ring stays in the camera frustum as a per-ring attribute, and use this information to determine the frame range needed for the simulation on every ring.

3. Points Scattering and Rendering

To create the highlights and details found in the fractal flame reference, we take the exported surfaces, scatter points (in the order of hundreds of millions) on them, resize every point to the width of a pixel, and allow overlapping points in screen space to accumulate in brightness as we render them. This results in areas of high density appearing with highlights in the folds and creases, which then fade off smoothly into thin, fine lines in areas of low density. (Fig. 6) For optimization, we calculate the screen distance size of each surface to determine how many points to scatter.

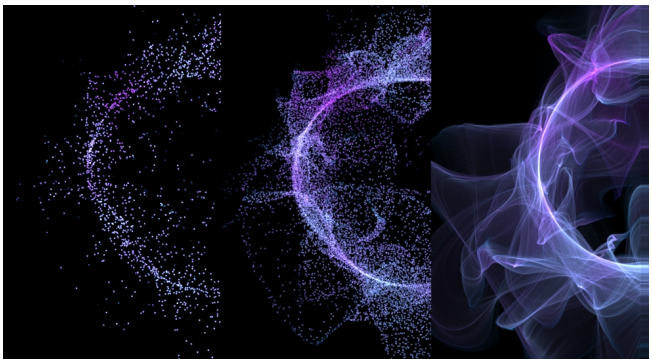
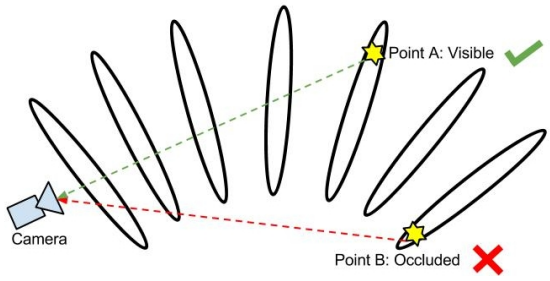


Figure 6. *Particles accumulate into fine lines*

To give the illusion of a continuous tunnel without the actual wall geometry, we need to occlude the points that appear behind the walls. This is tricky since these "walls" are defined implicitly. To solve this, we developed a reverse ray-marching technique: we convert the profile rings into planes and shoot rays from the render points back toward the camera. The rays need to hit all the planes consecutively between the point and camera to be considered "visible". If any one plane is missed, that source point is considered "occluded" and rendered darker, giving the impression of being outside the tunnel. (Fig. 7)



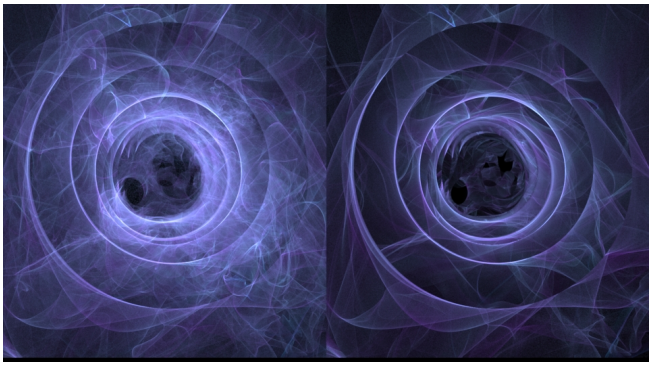


Figure 7. *Top: reverse ray-marching diagram. Bottom: before and after the occlusion treatment is applied*

There are several reasons why we chose to convert the surfaces into points instead of rendering the surfaces directly. The renderer we used is specialized in rendering massive amount of points much faster than rendering multiple overlapping and semi-transparent surfaces, and easily gives us the look that we were aiming for without needing to build any custom shaders. Furthermore, working with points gives us greater control to easily implement tunnel interior culling, camera frustum culling, and reverse ray-marching occlusion, all of which would have been more difficult to implement with surfaces.

A lot of shots have wormhole tunnels in the background. Since these tunnels do not require as much detail as the foreground ones, we use a faster technique to render them. We first simulate and bake out a generic set of undulating NURBS surfaces. Then, using the delayed load instancing feature from the Mantra renderer, we instance these surfaces along the background path curves and offset the animation of each instance by a random frame. This technique is both fast and memory efficient, allowing us to render many background flares quickly.

4. Compositing

To do color adjustments by the depth of the tunnel, we cannot simply change the point color in 3D space and render them every time since that is too slow. Additionally, because each pixel may be the result of multiple accumulated particles in depth, we can't rely on the depth channel from the render. Given all this, we instead render out a "depth map" of the tunnel by combining the inverse of all the profile ring planes and assigning a normalized depth value to each plane (Fig. 8a). Using this depth map as a mask in Nuke, the artists are able to do depth grading on the color by adjusting the lookup curves for the depth map (8.b). This method significantly reduces the turn-around time since the depth grading is done interactively at the compositing stage (8.c).

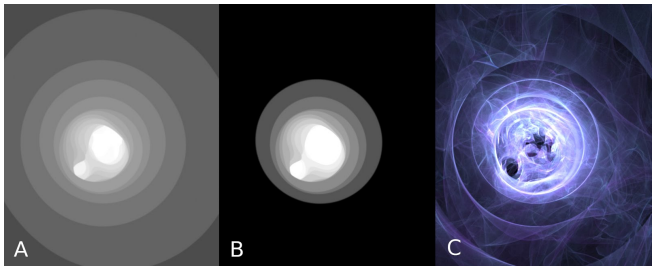


Figure 8. *Using the depth map for interactive depth color grading.*

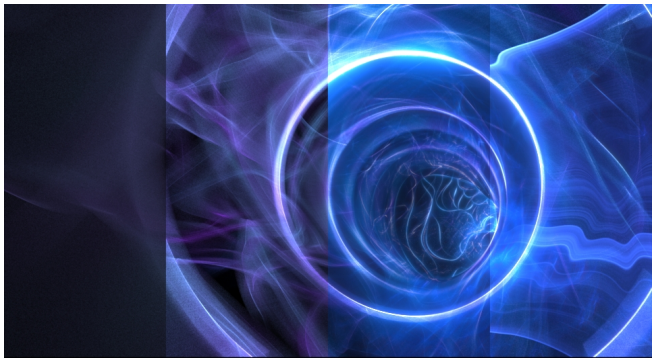


Figure 9. *Different stages in compositing*

Various other render passes are brought in to composite the final image. The occluded pass and the unoccluded pass are graded separately so the artist can differentiate the interior and the exterior of the tunnel. The profile rings are composited in to make the tunnels more well-defined. Laplacian operation is applied to further bring out the highlights in the render. Finally, optical tricks such as lens distortions and lens flares are added in to give the wormhole more life and energy.

5. Integration with Lighting

After the time travel wormholes are rendered and composited, we hand off the layers to the Lighting department so they can composite it with the time machine, which is rendered by Lighting. The time machine has a glossy exterior on which Lighting needs to reflect the wormhole tunnels. However, since Effects is rendering all the environment, we need an efficient way to hand those data off to Lighting for reflections. The traditional method of exporting point-based global illumination particles is insufficient since it does not produce enough visual fidelity. Instead, we use a cube mapping approach by rendering from six cameras that are parented to the time machine (Fig. 10.a), each with a 90 degree frustum facing an orthogonal direction. We then stitch the images together in Nuke to produce the reflection maps that Lighting can use. (10.b)

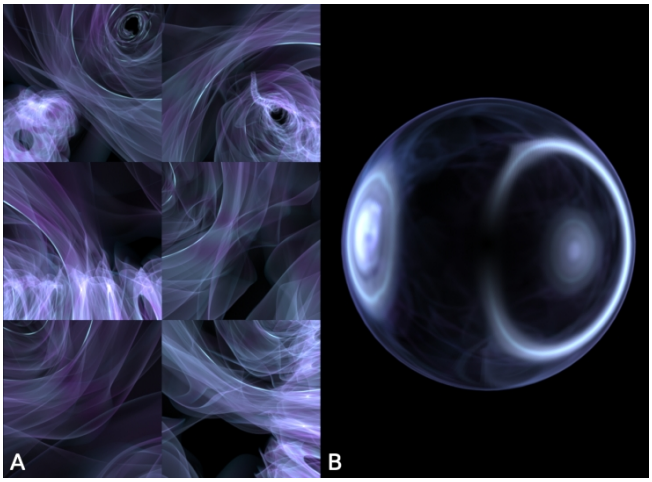


Figure 10. *Using cube mapping to produce the reflection map.*

6. Conclusions

By combining a mix of proven solutions, new techniques and optimizations, we developed a system that integrates multiple levels of complexity through several pipeline stages to create a unique and successful overall effect without actually using fractal formulas. Because of this, the effect is highly controllable, scalable, and allows various departments across the pipeline to work in unison. Some of the tools developed here, such as flares instancing and cube mapping, are further used in other effects such as the black hole sequence where thousands of flares have to be rendered and reflected on screen.

References

DRAVES, S. AND RECKASE, E., 2008. *The Fractal Flame Algorithm*. http://flam3.com/flame_draves.pdf. [Online; accessed 16-May-2014].