It's Raining Squids!

Simulating a rain of dissolving Squids for Watchmen

Mathieu Leclaire
Lead R&D, Hybride, A Ubisoft division
mleclair@hybride.com

Danny Levesque
Lead FX, Hybride, A Ubisoft division
dlevesque@hybride.com

Richard Clement-Tam
Lead Sequence FX, Hybride, A Ubisoft division
rclementtam@hybride.com

Christophe Damiano
Lead Lighting, Hybride, A Ubisoft division
cdamiano@hybride.com

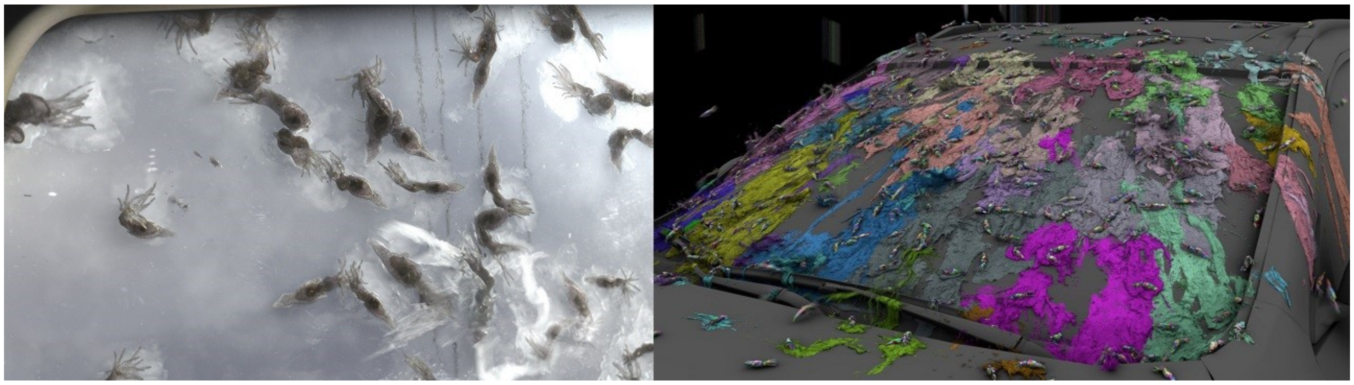

Figure 1: still image from Watchmen (left) & a capture of the simulation process (right). ©2019 Home Box Office, Inc. All Rights Reserved.

ABSTRACT

Simulating thousands of small squid-like creatures falling to their deaths, oozing a gel-like slime on impact and slowly melting away was no easy task. Managing the amount of pink one-eyed extraterritorial squids proved to be quite a challenge because they also had to interact with one another as well as with the different surrounding elements. We had complex fluid simulations to manage as the squids would first transform into gelatin and then completely dissolve. Shading the squids also proved to be quite a challenge because we had translucency and complex internal shadings to handle, which required very long render times since we were dealing with such a large quantity of creatures. We needed to be able to iterate quickly so we could work closely with the client in order to find that perfect balance of squid dynamics such as splashing, melting and shading.

CCS CONCEPTS

• **Computing Methodologies**; • **Physical Simulation, Animation and Rendering**;;

KEYWORDS

Simulation, deformation, rendering, Watchmen, FX, flip simulation

ACM Reference Format:
Mathieu Leclaire, Danny Levesque, Richard Clement-Tam, and Christophe Damiano. 2020. It's Raining Squids!: Simulating a rain of dissolving Squids for Watchmen. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '20 Talks), August 17, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3388767.3407357

1 INTRODUCTION

Setting the tone for this 2019 version of Watchmen, Hybride's most creative VFX work in Episode 1 translated into baby squids raining down on the city of Tulsa, Oklahoma.

The sequence refers to the Watchmen novel's ending, where a giant psychic squid is teleported into the center of New York City by Adrian Veidt on November 2, 1985. As Veidt predicted, world leaders believe the Earth is under attack by an alien threat and come together—ending the threat of nuclear war—to fight this new enemy.

As a reminder of this event, baby squids regularly rain down onto the city where residents seem relatively unfazed by the event. In this sequence, Hybride's extensive experience was deployed to create baby squids that change form completely once they've fallen from the sky.

2 SIMULATION

The first step consisted in figuring out how we could drop hundreds of little creatures from the sky and simulate contact with the ground as well as with the car where our main characters were seated. The first thing we tried was using Finite Element Method solver to simulate the fall and impact. To keep the desired level of detail required for each creature, we quickly realized we needed to use a high-resolution tetrahedrons mesh to properly simulate and deform our render mesh. The results of the dynamic simulations looked great, but we couldn't simulate more than a few at a time before the memory requirements exceeded the memory limit available in our machines. Since our mandate required that we simulate several hundreds of these creatures with full interactions, the FEM solver would be unable to manage the level of complexity we were looking for.

We decided to turn to Houdini's Vellum solver (a XPBD-inspired solver with substep-invariant stiffness) to see how far we could push the simulations. Although we obtained good results with faster simulation times and a little less memory foot-prints than with the FEM based approach, it still wasn't scalable enough to efficiently interactively simulate the quantity of squids we were trying to manage.

2.1 Instancing

For mid-range squids, we turned to instancing. We used FEM to simulate a few squids falling on a flat grid. Once we achieved simulations that were satisfactory, we cached those out and instanced them all over our scene with random orientations and time offsets. To obtain proper results on non-flat surfaces, we used a point based cage deformer to deform the actual simulations from a flat grid onto the curved surfaces where we instanced the squids.

We also mixed these results with particle-based simulations where we instanced pre-cached FEM impact deformations. We basically took the original FEM simulations and removed the translation to keep the squid at origin, while keeping the shape animations resulting from the collisions. We then synchronized and oriented the instances on the particles motion so we could match their impact time with the impact time of the zero out deformation cache.

2.2 Rigid Body Dynamics

For foreground shots where we see squids collide, results needed to be more precise so we turned towards a custom RBD solution. We created a lower-resolution outer shell of squids that we fractured using Voronoi. We then created soft constraints between the fractured pieces and used the template to simulate hundreds of copies. This way, we could easily and efficiently simulate several hundreds of RBD-squids. We just needed to create a few custom constraint forces to ensure the squids would stick to the collision surface, making sure they would bounce, roll and decelerate properly until we got a believable simulation.

Once we were happy with the simulations, we did a simple post-process to do a point based cage deformation of the high-resolution render meshes using the low-res fractured simulated meshes.

This meant we could now efficiently simulate all of our squids and have them interact with one another. However, the problem is that by using RBD, the squids started looking stiff and rigid, and lost the soft and squishy feeling you would expect from this type of creature. In order to retrieve our jelly-like property back into our simulations, we created a custom squish-and-squash deform operator that would compress the deformed mesh towards the surface using the surface normal with volume retention. The post-process enabled us to regain control and helped us deliver the expected results.

2.3 Splash Simulations

High-resolution FLIP simulations were required to produce gelatin-type liquids that splatter when the squids land on a surface. Covering the necessary surfaces needed to manage all of these squids at the high resolution we sought was impossible to simulate on one machine. We therefore created a system that would automatically cluster squids together to create smaller zones and we would then isolate the squids interacting in a zone and send a FLIP simulation on the farm so we could simulate each zone independently. The set-up would gather all the simulation caches and read them simultaneously to create a final render mesh. Although there was no interaction between the different zones because they were simulated independently, the overlapping nature of the results made it almost impossible to notice any lack of interaction. Having the simulations split into sections also meant that we would just have to simulate the area in question when changes were required. In this way, we didn't have to worry about resolution limits, which allowed us to quickly manage very complex simulations by dividing them among different machines.

2.4 Melting

The squids dissolved within 10-15 seconds after entering the atmosphere, but during the process they had to stay recognizable as long as possible.
In order to do so, we converted our squid meshes into a dense set of FLIP particles. We then used various propagating noises to increase the heat in the simulations affecting density and viscosity, allowing the mesh to melt off.

3 RENDERING

Raytracing through translucent creatures comprised of multiple complex internal structures can result in very long render times. Managing up to millions of them can clog your render farm if shading is not carefully managed.

Several controls were exposed to quickly activate / deactivate certain layers of shading and control the number of rays relative to motion blur intensity. We also encouraged artists to experiment with various denoising solutions to speed up renders. All these tools gave them enough flexibility to handle the complexity that comes with rendering multiple layers of transparency: reflection and refraction levels between the melting squids, the sticky residue from the previous squids; the wind-shield, and the actors in their environment location.

Lastly, we also needed to find a way to properly transfer the surface shaders and UV information to the FLIP particles in order to control the shading as the mesh deformed to ensure the textures didn't start to stretch as the squids were melting and dissolving.