# Taming the Swarm: Rippers on Pacific Rim Uprising

Martin Pražák
Double Negative, Lead RnD
map@dneg.com

Damien Maupu
Double Negative, RnD
dmu@dneg.com

Mungo Pay
Double Negative, RnD
mungo@dneg.com

Muhittin Bilginer
Double Negative, Lead FX artist
mub@dneg.com

Aleksandar Atanasov
Double Negative, FX artist
aata@dneg.com

Cristobal Infante Esquivel
Double Negative, FX artist
cies@dneg.com

**Ripper crowds in *Pacific Rim Uprising* were created using a combination of off-the-shelf crowd simulation in Houdini, and a number of in-house developed tools addressing specific needs. ©2018 Legendary and Universal Studios**

## ABSTRACT

When constructing shots of non-human crowds that exhibit complex behaviors, the standard approach based on the well-established rules of boid simulation is likely to fall short when used for a group of characters with "intent". In *Pacific Rim Uprising*, Double Negative VFX tackled the challenge of producing a large crowd of highly articulated robotic creatures performing the complex and coordinated task of "assembling" a mega-Kaiju. This task required a number of innovative approaches to both crowd authoring and rendering, and close collaboration between the RnD and artists.

## CCS CONCEPTS

• **Computing methodologies** → **Animation**; *Procedural animation*; Mesh geometry models;

## KEYWORDS

Crowd Simulation, Animation

**ACM Reference Format:**
Martin Pražák, Damien Maupu, Mungo Pay, Muhittin Bilginer, Aleksandar Atanasov, and Cristobal Infante Esquivel. 2018. Taming the Swarm: Rippers on Pacific Rim Uprising. In *Proceedings of SIGGRAPH '18 Talks*. ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.1145/3214745.3214809

## 1 BUILDING THE RIPPER

The ripper character in *Pacific Rim Uprising* was a robotic swarm creature resembling a biological organism. After a number of iterations, the final design is similar to an insect, but with tentacles instead of legs.

### 1.1 Modeling

The model of the character's body and tentacles comprised a large number of smaller semi-rigid parts, each with a number of "hooks" and other details to allow it to perform the intended actions.

To cater for different production use-cases, three levels-of-detail (LODs) of the mesh were built – the low LOD (17k polygons) was used primarily for viewport interaction during crowd authoring; the middle and high LODs (180k and 3M polygons respectively) were used for rendering, selected based on the distance to the camera. All LODs shared the same skeleton and procedural rig, allowing for the re-use of animation.

### 1.2 Rigging

The ripper's rig used a traditional combination of world and local space controls, with 622 procedural and animated joints driving the character's mesh via linear skinning. The skeletal framework required the tentacles to be represented as long chains of joints, making them hard to control using standard IK/FK techniques. To address that, each tentacle was built around a spline control with a number of sliding control points. This mechanism also allowed each tentacle to stretch and shrink, causing the separate rigid segments to slide in and out as required.

### 1.3 Animation

Despite an intuitive animation interface, the sheer complexity of animating a "walking" creature with 10 tentacles proved to be challenging. For this reason, all ripper performances in the movie were created using the crowd system, limiting the requirements for manual animation. Our animators were tasked to provide 3 groups of clips – *locomotion clips*, carefully synchronized to allow arbitrary blends; *actions* with precise transition points from and/or to locomotion clips; and a number of individual *poses* that could be layered on top of each locomotion clip.

## 2 CROWD

The movie sequence was split into three overlapping stages – first, a crowd of rippers was "flooding" the streets of a city, running towards the Kaijus; second, they formed "towers" to reach the Kaijus and moved on the surface of their skin; and third, they performed "cutting" and "stitching" actions, assembling a mega-Kaiju out of several smaller creatures.

Our crowd toolset of choice was SideFX Houdini™, with a number of customizations both in framework and user-interaction.

To represent the characters, animations and caches in the pipeline, we used a set of custom file formats, optimized for crowds. These translated to native representations in each respective DCC – native crowds in Houdini, animated meshes in the Isotropix Clarisse™ renderer and a set of custom display nodes in Autodesk Maya™.

### 2.1 Steering a ripper

Houdini includes a set of powerful frameworks for particle motion synthesis, varying in the underlying type of simulation. To give the artists the freedom to tailor different approaches to each scenario, we used particles to describe the behavioral part of the crowd simulation. In a separate crowd animation step, we then used the particle trajectories as steering targets ("carrots"). Finally, a boid-like "follow target" behavior drove the animation synthesis, assisted by a PD controller for faster response to direction and speed changes.

### 2.2 Animation synthesis

The animation of each ripper was produced using a combination of data-driven and procedural techniques.

The data-driven core was founded on parametric motion interpolation and blending. Local-space pose interpolation, used by default in Houdini's crowd tools, caused severe artifacts due to the complexity of the rig and its motion, and the length of the tentacle joint chains. With carefully designed animations, object-space blending (mathematically corresponding to mesh interpolation) produced significantly better results.

The baseline animation was then enhanced by procedural effects – the agent's speed influenced the amount of clip timewarping and the amplitude of dynamic tail swing, while the agent's angular velocity was used to derive appropriate tail and body bending.

Motion trees provided motion transitions, with timings derived from artist-driven events. The action-packed shots were all short so it was never necessary to transition more than once for each character, simplifying the motion planning.

### 2.3 Street level

On the street level, the client's requested a "flood" of characters, with appearance akin to a fluid simulation rather than a swarm of insects.

To address this requirement, we based the crowd behavior on a number of FLIP simulations, with driven particles ("grains") determining the character trajectories. These were then layered on top of each other to provide a "shifting" effect, smoothed out to remove any jitter, and used as targets for the crowd animation synthesis. The grain simulation also inherently handled the fixed minimum distance between characters.

To enforce environment contacts ("tentacle-roll"), we designed a custom IK cyclic coordinate descend (CCD) solver. However, due to the visual complexity of the scene, it was not required.

### 2.4 Creeping rippers

During the third stage of the Kaiju assembly, rippers were clinging to the surface of each moving creature, performing both locomotion and specific cutting/sewing actions as directed by the artists.

While the creature's surface deformed between each frame, its topology remained consistent. By representing all particle data using barycentric coordinates and using this topological consistency to move particles between polygons, we were able to both constrain the particles on its surface, and transfer data between frames.

The artists' directions were represented in two ways – via a set of *guide curves*, and a set of *task points*, both tied to the underlying mesh. Guide curves were evaluated into a continuous direction field, which influenced the direction of each target particle; task points acted as distance-triggers, activating animation transitions.

The same approach was used to create "towers" of rippers, reaching for the Kaiju during the second part of the sequence. Each tower had an underlying set of layered animated guide geometries, allowing the artists to precisely sculpt the performance of the swarm.

### 2.5 Strands

The strands of rippers performed a major part of the "stitching" behavior in the sequence. Their performance was based on two steps – first, the rippers were animated on a cylinder of appropriate length at the origin, and their position and orientation was recorded in cylindrical coordinates. Second, a wire simulation provided the overall target shape of a strand, and each ripper was transferred from its original position to a position relative to the deformed curve.

The strands themselves consisted of a number of "bands", with each band containing rippers performing a specific animation (e.g., surface interaction, different types of inter-locking). The characters in bands at each end of the strand were constrained to the animated surface, providing the anchor points.

## 3 RENDERING

The sequence was rendered using a modern path tracer in Isotropix Clarisse™. Path tracing produces results that are more physically correct than those produced by older object-based methods. However, the unpredictable nature of ray distribution requires the whole scene to be contained in the main memory.

Naively representing each character as a separate mesh, a method we use for human crowds, proved to be extremely memory-intensive. Instance scattering, usually possible for "stadium-type" scenarios, would severely constrain behavior complexity. Mesh de-duplication was excluded by the fact that each character is procedurally articulated, and there is little similarity between different character poses in a single frame.

To obtain instantiable rigid objects, we used skinning weights to split the mesh in the renderer to a number of segments ("calamari"). This allowed us to render 6,000 detailed characters – a number that would significantly exceed the memory limits without instancing.