Making Waves for Surf's Up

Rob Bredow Daniel Kramer Matt Hausman Peter Shinners Deborah Carlson John Clark
Erick Miller
Sony Pictures Imageworks

For the Sony Pictures Animation feature film, Surf's Up, Imageworks needed to create a realistic hero CG ocean and wave system. To create approximately half of the shots in our film, we required the look and feel of real masses of curling, breaking water, all the dynamic aspects of crashing breaking whitewater and turbulent surface ripples, foam across and over the face of the wave – yet still it needed to be interactively used in animation to compose camera angles, define shot lengths and allow 100% interaction with fully animated surfing characters.

1 Solving A Fluid Problem

Fluid dynamics and fluid simulation based approaches, as techno-appealing as they seemed, were quickly ruled out due to their processor intensive simulation based nature. Full predictability of the fluid surface produced for animators was needed to create consistent key framed animations along with fully animated CG characters. The wave was broken down in an analytical and procedural layered approach, thus decoupling it from time based methods and, as a result, made it a hero CG character instead of an effects element of the film. Ultimately, a hybrid system was built, manipulated virtually in real-time by artists and animators. Shots could be built and composed based on size and speed of water, allowing accurate in-camera previews for surf moves to be skillfully maneuvered across the faces of the foaming waves in a fast, predictable, and realistic manner - and later using the animation to analytically interpolate dynamic information used for advanced shaders and effects applied to the final rendered fluid surface.

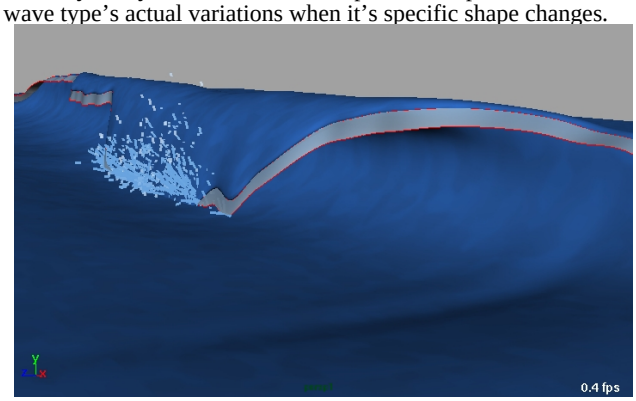
2 Anatomy of a Procedural Wave System

The decision for the wave to be a character meant that we determine, classify and create controls that procedurally drive something as complex as a fluid surface with only a handful of simplified interfaces exposed to the animators. We broke this down into three very general categories

1. Predefined 2-D wave profiles create a 4-D sampled surface interpolation, with the third dimension being the length parameter, and the fourth dimension being decoupled time.
2. Interface controls to modify and animate surface based fluid interpolation are simple UI manipulators that interactively drive the fourth dimension of decoupled time, along with sampled shape parameters that modify interpolated profiles.
3. Dynamic texture, water surface, wake trail and splashing, crashing real-time particle preview engines that are also able to be decoupled from their time dependencies.

3 Riding The Wave

The first thing to remember while creating this fluid system was that animators would be using it every day to set keyframes and to surf around on. A special geometric constraint was created within the published wave character called a wave rider, which allowed the user to freely move a surfboard across the wave automatically staying "above water" but not distorting as the wave face itself changes shape and shifted coordinate values. In an attempt to simplify & consolidate controls, a user interface was designed called the "Wave Wizard," where an animation library of surfer approved wave animations, as well as the switching and preview controls for the wave modes, could be accessed by the user. Animation attributes themselves - allowing creation of all the variations of wave type controls, were logically named, such as "pipeline," "mavericks," "spilling breaker," etc. The activation of these seemingly simple attributes, though, set in motion a large array of settings that not only changed the appearance of the wave, they changed surface interpolation parameters, look-dev settings, whitewater crash settings, as well as a very large array of predefined data sets that feed the wave node its data for the algorithmic interpolation scheme. Some of the finer typed controls shared across all wave types were lip up/down, forward/back, trough depth, shoulder size, tube depth, tube length, front and back length, non-uniform scale. The customized per-wave type shape controls were also time interpolated, created and carefully analyzed to resemble sampled time splices of the each wave type's actual variations when it's specific shape changes.



4 Interpolating A Fluid Solution

4-D surface interpolation was exclusively handled using spline interpolation. Initial experiments were done with rotation based approaches, but in the end the spline result was very appealing because it was an efficient interpolation scheme, and yielded a nice fluid curve tangent that guaranteed to interpolate directly through actual data being sampled. Finally, the wave generation interpolation algorithm not only output the surface, but also attached itself to any arbitrary value assigned to the incoming wave profiles – this allowed any wave to later have predefined attributes such as "energy," "time," "speed," "crash," etc. all which would automatically be output as interpolated per vertex gradient values according to what they were defined as on the time sampled input profiles.

5 Ocean and Water Previews

The Wave Wizard UI contained an option for previewing texture or particle previews, called Wave Blast, which would enable certain controls on the rig and write out a flip book of images. Texture and speed preview modes were available via in-rig hardware shading which previewed the interpolated texture reference space, so that the animator had a pixel accurate representation of the animated texture space pre-render, which created the appearance of water flowing over the surface. The ability to visualize the wake trail coming off the back of the surf board was built in, via wave riders and a time based lookup script using a procedural geometric wrapping technique. The rig contained ray intersection along the lip, called the crash curve, which was interpolated with quaternions of "spill vectors" to determine its ray direction, and then output interpolated energy data into an open GL particle preview node which displayed real-time preview of crashing white water particles in-scene. Finally, the actual surface turbulence of the water could be previewed within the rig via a custom deformer called the Wave Train node that read in animated displacement textures and performed a vertex displacement that matched accurately to the displacement shader used to render the ocean water, all within an animator's scene.

e-mail: erickmiller@yahoo.com