From a calm puddle to a stormy ocean - Rendering water in *Uncharted*

Carlos Gonzalez-Ochoa * Doug Holder Eben Cook
Naughty Dog, Inc

Introduction

The Uncharted series of video games for the PS3 have been recognized by their distinct cinematic gaming experience and high quality graphics. Although the core game mechanics are centered around exploration and combat; and less on water game play, water has still been a major design element of the game. The complexity of the game mechanics and rendering of the water has increased over the three games in the series. The range of water types goes from simple puddles, pools to lakes and rivers, and, finally an ocean storm environment. The games art style is realistic, but not fully photo-realistic and the water rendering has to match the style. Our water shader uses general optical principles but adds artistic controls to realize the style of the game. One of the biggest challenges we solved was processing and rendering water with dramatic movement and striking shading while using a small portion of the available resources (memory, SPU and GPU). The water system runs in parallel with other game systems and everything is running at 30 fps. In Uncharted 3, the water system was used to drive a cruise ship, in which the whole level resided and thus became a central element of the gameplay. The water would drive the cruise-ship which in turn moves the player, enemies, effects and physics objects. In addition there were levels with swimming in rough water, flooding water, crashing waves and floating platforms that had their own technological challenges. Our engine uses different render engines depending on the type of water. All of these engines are procedural systems and, because of the game design, not one uses real-time physics simulations. One renders non-LOD meshes with calm and semi-moving water bodies (rivers, lakes, and puddles), another uses a hierarchical LOD system with displacement for open bodies of water, like the ocean, and the last uses a skinning mechanism with particles for the flooding events.

Shader

To achieve a stylized look for the water we use a complex shader with multiple controls of refraction, reflection, foam and a depth based effect (called churn) to give the water a volumetric effect. The most important feature is flow, which gives a distinct water movement. The movement is created by advecting the triangle's or pixels uv coordinates of the normal maps [1996]. We blend two or more normal maps, that are offset in phase, resulting in a continuous movement effect. A surface wide vector field is used to define the local direction of flow (per triangle or pixel), we also use other maps to determine displacement magnitude, bump strength, foam placement and other properties. A feature of real life rivers that we wanted to achieve are stationary or semi-stationary waves, which are different from the rolling, curling waves of oceans. For such, we displace vertices in a circular motion on a vertical plane parallel to the flow direction. The vertices are animated in SPU processes before they are sent to the GPU for rendering.

Open Ocean rendering

The ocean is rendered as a mesh using a hierarchical LOD system using an alternate scheme of Geometric Clipmaps [2004]. The mesh is composed as a series of concentric rings of polygons centered from a point of interest close to the camera center; and each ring has successively coarser resolutions. Our scheme differs by how each ring is divided into different sized patches. This improves

culling and visibility, balance SPU job loads and guarantees continuity and blending across ring levels to avoid T-junctions. Each vertex of the clipmap will later be displaced using a procedural wave system. The displacement acts in every direction not only in the vertical so we can achieve sharper wave peaks. The LOD generation and wave displacement is completely procedural and runs in several SPUs in parallel. The system is procedural, parametric and deterministic. It can be used to generate the renderable mesh or to drive game objects.

To simulate the wave motion of an open ocean we generate a field of wave particles [2007]. This method was chosen instead of the well-known FFT or procedural noise methods, because it provided more intuitive controls to artists, didnt exhibit any tiling artifacts at low grid resolutions and was simple to optimize in the SPU. We add low frequency time modulated Gerstner waves to define a general look and use an extruded NURBS curve profile to create specific wave shapes. In addition, the motion of the waves displacement drives the flow and foam shader parameters. For collision queries (point and ray tracing) we dont perform tests against the rendered mesh, but instead use a search method on the displacement field, which gives faster and more accurate results. We compose the final wave displacement as a composition simpler waves and displacement grids. Our artists can add several grids at different scales to create octaves and create wave detail at different frequencies. These controls are flexible and intuitive for artists to use and expresive enough to create a calm swimming pool to a stormy ocean.

Flood

For the flooding events, we used a combination of skinned meshes driven by the result of an offline simulation and artfully placed particle effects. The moving water was simulated offline to generate a high resolution mesh. From it a series of lower resolution animated meshes were created using a shrink-wrapping tool that was custom built in Houdini. The resulting meshes were guaranteed to have a defined vertex count that could be animated in the game engine at run-time. The simulation was also used by the artists as a guide for the placement and timing of particles. To limit overdraw particles are rendered in half-screen buffers.

References

LOSASSO, F., AND HOPPE, H. 2004. Geometry clipmaps: Terrain rendering using nested regular grids.

MAX, N., AND BECKER, B. 1996. Flow visualization using moving textures.

YUKSEL, C., HOUSE, D. H., AND KEYSER, J. 2007. Wave particles.

*cgonzoo@gmail.com, doug@dougvfx.com, supereben@gmail.com

SIGGRAPH 2012, Los Angeles, California, August 5 – 9, 2012.
ISBN 978-1-4503-1435-0/12/0008