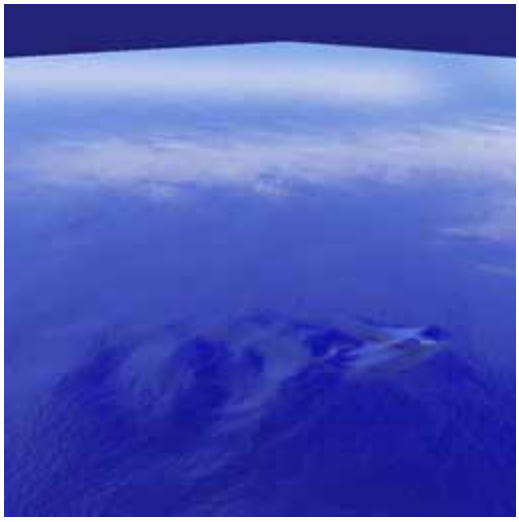
Technical Report

Water Interaction



Water Interaction

This sample simulates and renders an animated water surface using programmable graphics hardware supporting Microsof Direct3D Vertex and Pixel Shaders 1.1 and later. The techniques used here are useful in a wide variety of applications and are:

1) Using render-to-texture and programmable pixel shaders to calculate a physics simulation on the GPU.

2) Using pixel shaders to convert a height-map into a normal map or bump map for environment maped bump mapping. This efficiently creates an animated bump map in video hardware.

3) Coupling two dynamic water simulations together so that a single tiled texture can transition seamlessly to a local unique detail texture. Interactions from local objects and characters can be applied to the detail texture.

4) A method of rendering water reflections similar to Environment Mapped Bump Mapping (EMBM) but using vertex shaders and the pixel shader texm3x2tex operation to vary the EMBM rotation matrix from vertex to vertex. This allows the displacement direction to vary across the screen and to fade out with increasing distance.

The coupling of two water simulations allows a large area of procedural water to be created while still allowing for unique local features. One simulation texture tiles seamlessly, so it can be repeated across a large area. The borders of another simulation, the “local detail” simulation, match the tiled texture, and the interior of the local detail texture can evolve separately acording to whatever unique local perturbations are applied. This allows for localized features in the water that don’t tile across the large area. Displacements from a character can be rendered into the detail texture, and they will fade out toward the edge until they transition seamlessly to the tiled texture. The detail texture is free to move around to follow the character.

For reflections off the water surface, the same calculation for DX6-style EMBM is performed in a custom pixel shader. The displacement to a base texture coordinate after a 2x2 matrix transform can be computed in a Direct3D pixels shader 1.1 program with the texm3x2tex instruction. This performs two dot products – one to calculate the U texture coordinate, and the other to calculate the V coordinate. The 2x2 transform is supplied to the pixel shader through interpolated vertex texture coordinates, so the matrix can vary from vertex to vertex. This is an improvement to the fixed-function EMBM operation, since the fixed-function method uses the same 2x2 matrix for all pixels. By varying the matrix per-

vertex, the bump mapping can vary across the screen or fade out with distance. The texm3x2 instruction involves two three-dimensional dot-products, so the reflection method is called "dot3x2 EMBM."

This sample presents a different method for environment reflection than the traditional EMBM or cube map reflection methods. The EMBM calculation is used to calculate reflection into a sky dome environment map, rather than into a flat reflected view of the scene. This allows the reflections to sample from the entire hemisphere above the water surface, instead of sampling from a small local area as with traditional EMBM. The sky dome environment map is easy to render on the fly. It is simply a very wide-angle view looking straight up from some point on the water surface. This environment map will not be able to corectly reflect objects near the water, such as overhanging trees or characters in the water. Cube environment mapping has the same problem, in that the reflection map is computed from a single point and does not vary from vertex to vertex.

The C++ class that drives the water simulation can generate a variety of bump maps for various reflection techniques. It can generate a normal map for cube environment mapping, a displacement map for traditional EMBM, or a bump map for the dot3x2 EMBM reflection which is similar to a normal map but where the blue channel is always 1.0.

The water simulation is performed entirely on the graphics hardware using pixel shaders. When rendering each pixel of the simulation, the shaders sample each pixel's neighbors and compute the next time step based on the neighbor's values. The physical state of the water is stored in a single A8R8G8B8 texture with one texel for each point of the simulation. The colors of this texture hold the height, velocity, and force acting at each texel. Height is kept in the blue and alpha colors, velocity in the green, and force in the red. There are two texture render targets used in a ping-pong fashion, where the previous frame's texture result is used to render to the other texture render target. In this way, rendering back and forth between textures produces an endless non-repeating simulation of rippling water. Displacements can be rendered into the water simulation, and they will spread out into naturalistic ripples and waves.

The process of updating the height takes two render-to-texture passes on pixel shader 1.1 hardware. The first accumulates the force contribution from three neighbors of each texel. The second adds the force from a fourth neighbor, then applies the total force to the velocity, and then applies the new velocity to the position. The new position and velocity are output as an RGBA color. A third pass smoothes neighboring positions and velocities in order to reduce undesirable high frequency oscillations.

Bibliography

1. Gomez, Miguel, "Interactive Simulation of Water Surfaces" Game Programming Gems, ed. Mark A. DeLoura, Charles River Media, Inc., 2000, pp.187-199 ISBN: 1-58450-049-2
2. Greg James, “Operations for Hardware-Accelerated Procedural Texture Animation” in “Game Programming Gems 2” ed. Mark DeLoura, Charles River Media : 2001, ISBN 1-58450-054-9, p 497
3. Lengyel, Eric, "Mathematics for 3D Game Programming and Computer Graphics" Charles River Media, 2002, p. 327 ISBN: 1-58450-037-9
4. NVIDIA Developer Web site: 2003. http://developer.nvidia.com

Notice

Trademarks

Copyright

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com