A GPU-Accelerated Hydrodynamics Solver For Atmosphere-Fire Interactions

Abstract
By studying scaled-down models for fire dynamics, we are investigating fundamental processes that are critical to fire spread. One of the most important factors is the complex coupling of the atmospheric flow and the combusting environment. In particular, what variables influence fire induced flows, and how do they contribute to fire spread? On the technology side, we are developing physics-based hydrodynamic models with a high degree of data parallelism that is ideal for graphical processing units (GPUs). The dynamic meshing visualization used in OpenGL in an augmented reality environment.

Methods (cont.)
GP-GPU Programming
CUDA (Compute Unified Device Architecture) is a parallel computing technology developed by Nvidia for graphic processing. CUDA programming environment allows the development of general-purpose GPU programs for the implementation of numerical calculations on GPUs.

Heterogeneous Programming
CUDA programming model that consist of CUDA threads executing on a physically separate device (GPU) and operating as a represlookor to the host (CPU) running the C++ program.

Results
Interactivity will be incorporated with a 1:1 real sand table filled with reflective sand. A projector displays output visuals from the connected computer onto the table, and a Kinect camera determines the topography. The simulation will be rendered as a topographical model to visualize point-source spot fires spreading across a terrain and adding various variables. Hand gestures/motions will manipulate the simulation and the terrain can be altered in real-time by moving the structure of the sand.

Future Work
Interactive simulations will be executed on a physically separate device (GPU) and operating as a represlookor to the host (CPU) running the C++ program.

References & Acknowledgements

Acknowledgements:
Florida State University, Department of Scientific Computing
Dr. Bryan Quaife, Dr. Kevin Speer

Implementation

addSink ()

advectParticles ()

diffuseProject ()

updateVelocity ()

addForce ()

addSink ()

advectParticles ()

addForce ()

The velocity fields, vxfield and vyfield, will be used in the next routine.

vxfield and vyfield (each having their own x and y components for FFT). Modifications from the previous fields are applied at the projection step.Inverse Fourier transforms applied to the real fields.

w_i(x) = w_i(x) + \Delta t w_i(x).

w_i(x) = w_i \cdot \left(1 + \frac{\Delta t}{\nu_j} \right) w_i(x).

w_i(x) = \frac{w_i(x) - \frac{\Delta t}{\nu_j} \int \int \int \nu_j \delta(x-x_0) dx dy dz}{w_i(x) + \Delta t w_i(x)}.

The newly projected fields are normalized and combined back to the device field.
The velocity fields, vxfield and vyfield, are added to the velocity field.

part(t + 1) = part(t) + dt \cdot \int \int \int \nu_j \delta(x-x_0) dx dy dz.

The final result takes the device field as a parameter and updates the particles by moving positions according to the velocity field and time. Then each particle is mapped to a texture object and bound for visualization.