AMD **Accelerated** Parallel Processing TECHNOLOGY

SAMPLE

2D Fluid Simulation Using the Lattice-Boltzmann Method

1 Overview

1.1 Location

`$(AMDAPPSDKSAMPLESROOT)\samples\opencl\cl\app`

1.2 How to Run

See the *Getting Started* guide for how to build samples. You first must compile the sample.

Use the command line to change to the directory where the executable is located. The default executables are placed in `$(AMDAPPSDKSAMPLESROOT)\samples\opencl\bin\x86` for 32-bit builds and `$(AMDAPPSDKSAMPLESROOT)\samples\opencl\bin\x86_64\` for 64-bit builds.

Type the following command(s).

1. `FluidSimulation2D`
   This runs the simulation in a 256x256 grid.
2. `FluidSimulation2D -h`
   This prints the help message.

1.3 Command Line Options

Table 1 lists, and briefly describes, the command line options.

Table 1 Command Line Options

Short Form	Long Form	Description
`-h`	`--help`	Shows all command options and their respective meaning.
	`--device`	Devices on which the program is to be run. Acceptable values are cpu or gpu.
`-q`	`--quiet`	Quiet mode. Suppresses all text output.
`-e`	`--verify`	Verify results against reference implementation.
`-t`	`--timing`	Print timing.
	`--dump`	Dump binary imge for all devices.
	`--load`	Load binary image, and execute on device.
	`--flags`	Specify compiler flags to build kernel.
`-p`	`--platformId`	Select platformId to be used (0 to N-1, where N is the nuber of available platforms).
`-d`	`--deviceId`	Select device to be used (0 to N-1, where N is the nuber of available devices).
`-i`	`--iterations`	Number of iterations for kernel execution.

2 Introduction

This sample is based on the Lattice-Boltzmann method (LBM) for simulating a fluid on a 2D grid. The format is D2Q9, which represents the second dimension and nine frequency distributions of velocity. Each cell has nine velocity directions, with indexes shown in Figure 1 (not necessarily in the same order as the sample).

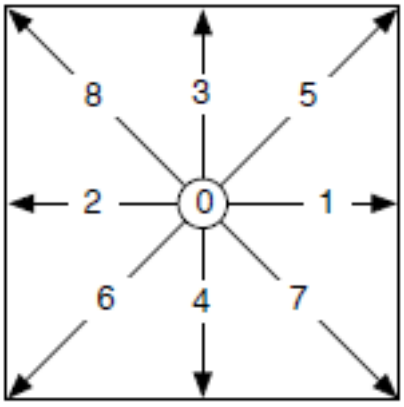

Figure 1 Velocity Directions

The velocity and density values of a cell can be calculated by using the equations

$$\rho = \sum f_i \quad u = \frac{1}{\rho} \sum f_i \, e_i$$

where

f_i is the frequency of each velocity (the number of particles in a cell going in that particular direction).

i represents values from 0-8, inclusive, denoting a particular velocity, as shown in Figure 1.

e_i denotes the direction vector of the velocity.

The Lattice-Boltzmann equation can be written with respect to space and time as

$$f_i\,(x,t + \Delta t) - f_i(x,t) = \Omega_i$$

where

Ω_i is a fluid collision operator that models the collision between fluid molecules in a cell.

We use a simple collision operator, called the Bhatnagar-Gross-Krook approximation, which uses a single relaxation time approximation to reduce the operator to operations suitable for computers. It is based on the idea that the main effect of the collision operator is to bring the molecule distribution closer to the equilibrium distribution, which is defined by

$$f_i^{eq} = \omega_i \rho \left(1 - \frac{3}{2} u^2 + 3(e_i \bullet u) + \frac{9}{2}(e_i \bullet u)^2\right)$$

where

ω_i is a constant that depends on the lattice geometry.

The collision operator is defined as

$$\Omega_i = -\frac{\Delta t}{\tau}\left(f_i(x,t) - f_i^{eq}(p,u)\right)$$

where

τ is a constant that represents the viscosity of the fluid.

Combining the equations

$$f_i(x + e_i, t + \Delta t) = f_i(x,t) - \frac{\Delta t}{\tau}\left(f_i(x,t) - f_i^{eq}(p,u)\right)$$

The basic steps of the LBM are:

1. Iteration start – Initialize the values of the velocity distributions for each cell.
2. Collision step – Compute the density and scalar velocity of a cell, and calculate the equivalent distribution.
3. Streaming step – Distribute the newly computed frequency distribution (according to final equation) to neighbors.

The diagram in Figure 2 explains the 3 steps.

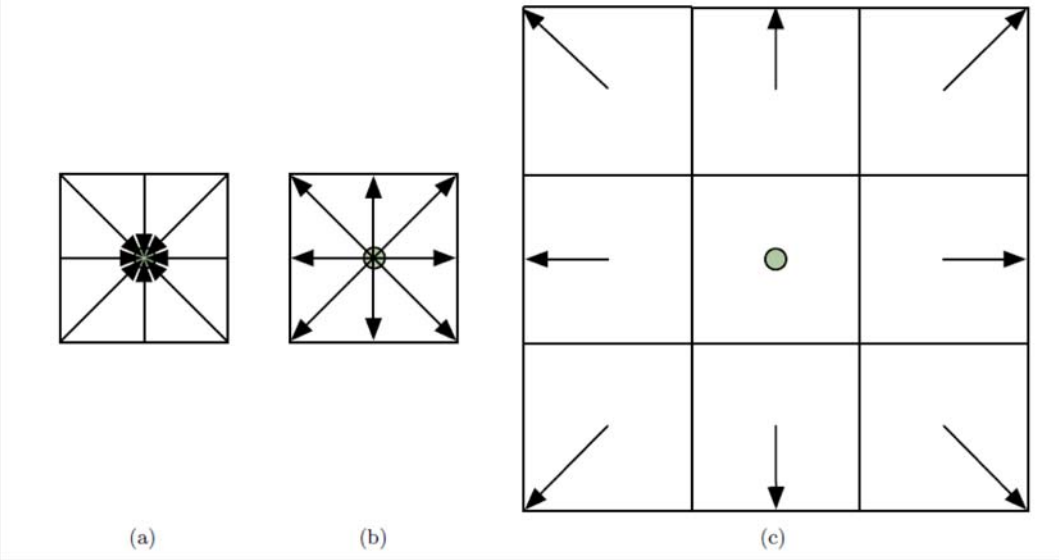

Figure 2 Basic LBM Steps

3 Implementation Details

The simulation is initialized and updated by holding the left-mouse button down while dragging the mouse across the entire grid. This assigns velocity values to the cells, and the simulation continues from there.

There are two sets of buffers: one for the input frequency distribution, the other for the output frequency distribution. These store nine values for each cell in the grid. There is one velocity buffer which stores the resu ltant velocity for each cell. There is also one type buffer, which differentiates between empty space and boundary cells. The boundary cells are created dragging the right-mouse button on the grid.

In each step, the kernel updates the output buffer with the new distribution, which becomes the next input. The velocity buffer is translated to RGB values using a color conversion table, and this buffer is displayed using a GL renderer.

The sample uses double precision, so it runs only on following devices: Radeon 4800 series, Radeon 5800 series, and CPU. The color conversion is done on the host; thus, with every step buffers are copied to the host. As a result, the GPU might process fewer frames per second than the CPU.

4 References

1. The OpenCL code is based on the openMP code available at http://software.intel.com/en-us/courseware/course/view.php?id=87
2. The theory presented is taken from http://www.monitzer.com/FluidGPU/

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For AMD Accelerated Parallel Processing:
URL: developer.amd.com/appsdk
Developing: developer.amd.com/
Support: developer.amd.com/appsdksupport
Forum: developer.amd.com/openclforum