CS 5220 Introduction to Parallel Programming Fall 2015
Kenneth Lim (kl545), Scott Wu (ssw74), Robert Chiodi (rmc298), Ravi Patel (rgp62) Final Project

Introduction

Smooth Particle Hydrodynamics (SPH)

Purpose

The purpose of this project is implement a Smoothed Particle Hydrodynamics based on Position Based Fluids (Macklin, Muller 2013). Since the goal of the paper is geared towards real time use, some portions of the algorithm will sacrifice accuracy for speed. Finally, we will be analyzing and parallelizing the algorithm to further improve performance.

Algorithm Overview

In the simulation, our inputs are the initial positions and velocities of particles in a box. Then we take equal time steps, producing the same list of position and velocity outputs at each step.

- At the beginning of each step, we compute candidate velocities and positions by taking an Eulerian step
- We compute the neighbors particles within a certain radius by placing them on a grid
- Candidate velocities and positions are iteratively corrected
  - Per iteration, we attempt to solve the incompressibility constraint
  - Meanwhile maintain the boundary conditions of the box, and apply velocity dampening if necessary
  - Update the candidate positions with those constraints
- Using the new candidate positions, we update the candidate velocities
- We apply vorticity confinement to maintain energy in the system
- We apply viscosity to blur the velocities into a more coherent motion
- Finally we update the positions and velocities with the candidate positions and velocities

Code Bases

C Code

Fortran Code

Profiling and Serial Optimization

C Code

Profiling

Optimizations

Compiler Flags

Fortran Code

Profiling

Optimizations

Compiler Flags

Parallelization

C Code

OpenMP

Case	# Particles	LLC (x,y,z)	URC (x,y,z)	# Threads	Time (s)	SS (%)	SSE (%)
1	36,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	1	182.99		
2	36,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	2	79.769		
3	36,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	4	42.479		
4	36,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	8	24.816		
5	36,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	12	17.791		
6	36,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	16	22.772		
7	36,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	20	18.565		
8	36,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	24	28.774		

[tab:ss]

Case	# Particles	LLC (x,y,z)	URC (x,y,z)	# Threads	Time (s)	SS (%)	SSE (%)
1	2,197	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	1	6.3919		
2	4,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	2	7.2487		
3	8,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	4	10.717		
4	16,000	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	8	11.526		
5	24,389	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	12	14.846		
6	32,768	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	16	24.198		
7	39,304	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	20	24.268		
8	46,656	(0.0, 0.0, 0.0)	(0.5, 0.5, 0.5)	24	24.203		

[tab:ws]

Strong Scaling:

Weak Scaling:

Cilk

Strong Scaling:

Weak Scaling:

Fortran Code

OpenMP

Strong Scaling:

Weak Scaling:

MPI

Strong Scaling:

Weak Scaling:

Summary and Future Work