Implicit FEM and Fluid Coupling on GPU for Interactive Multiphysics Simulation

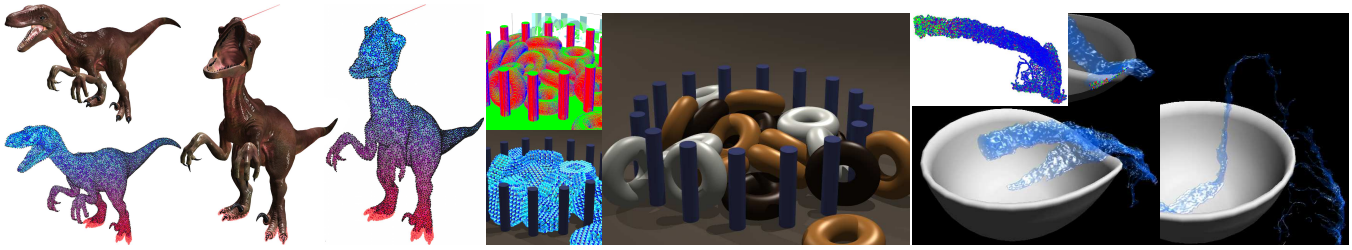


Figure 1: *We combine GPU methods for detailed deformable objects (left), image-based collisions (middle), and SPH fluids to simulate a two-way fluid-deformable coupling (right) at interactive rates.*

Abstract

We present a method to implement on the GPU an implicit FEM solver which is fast and stable enough to handle interactions and collisions. We combine this method with GPU-based fluids [Zhang et al. 2008] and collision detection [Allard et al. 2010] to achieve interactive multiphysics simulations entirely running on the GPU.

1 Introduction

The Finite Element Method (FEM) is widely used to simulate deformable materials in mechanical simulations. Recently co-rotational linear FEM was applied successfully in interactive games [Mendoza and Garcia 2010] and medical simulations, however CPU methods are limited to coarse meshes due to performance constraints. GPU-based FEM methods have been proposed [Comas et al. 2008], but they rely on explicit schemes and thus require prohibitively small time-steps. We overcome these limitations by implementing an implicit integration scheme on the GPU. GPU collision handling methods for rigid [Tonge et al. 2010] and deformable bodies [Allard et al. 2010] are available, yet an interactive simulation of two-way fluid-deformable coupling is an open problem.

2 Methods

2.1 Implicit FEM Solver on GPU

To implement implicit FEM, we rely on an iterative Conjugate Gradient (CG) solver. However, in contrast to existing GPU-based sparse solvers [Krüger and Westermann 2005; Buatois et al. 2009], we do not explicitly build the system matrix, but instead parallelize the required matrix-vector products directly on the original mesh. This considerably reduces the number of operations required, and more importantly the consumed bandwidth, enabling the method to be fast enough for interactive simulations of soft bodies. The parallelization, detailed in [Allard et al. 2011], relies on first computing the contribution of mesh elements using one thread per tetrahedron, followed by a parallel gather to accumulate contributions at vertices. Further optimizations include mesh ordering, compact data structures, memory layout, and changing sequences of operations to reduce synchronization points.

2.2 Fluid Coupling using Image-based Collisions

An image-based collision method [Allard et al. 2010] has been proposed to handle complex deformable objects. It computes intersection volume gradients which are discretized on pixels and accumulated on vertices. To handle fluid coupling, we extend this approach to compute pixels directly from the SPH fluid density field using ray-tracing. When intersections are detected, contributions are accumulated to the fluid particles based on their relative contribution as given by the SPH kernels evaluated at the pixel. As ray-tracing can be expensive, an important optimization is to test rays only when an existing pixel is within the fluid, which requires only a simple evaluation of the SPH density field.

3 Results

All timings given below exclude rendering as the cost of this step can vary greatly depending on the desired visual quality and complexity. Our CUDA-based FEM solver is able to simulate a deformable object with 45k tetrahedral elements (Fig. 1 left) at 212 FPS on a Nvidia GeForce GTX 480, 18× faster than our most optimized sequential implementation on an Intel Core i7 975 3.33GHz CPU. The fluid simulation (Fig. 1 right) demonstrates two-way coupling between the fluid and a soft cup, achieving 25 FPS using 2k FEM elements and 32k SPH particles on a GeForce GTX 280 GPU.

References

ALLARD, J., FAURE, F., COURTECUISSE, H., FALIPOU, F., DURIEZ, C., AND KRY, P. G. 2010. Volume contact constraints at arbitrary resolution. *ACM Trans. Graph. 29*, 4.

ALLARD, J., COURTECUISSE, H., AND FAURE, F. 2011. Implicit FEM solver on GPU for interactive deformation simulation. In *GPU Computing Gems Jade Edition*. Elsevier. to appear in June.

BUATOIS, L., CAUMON, G., AND LÉVY, B. 2009. Concurrent number cruncher - a GPU implementation of a general sparse linear solver. *Int J Parallel Emerg. Distrib. Syst. 24*, 3, 205–223.

COMAS, O., TAYLOR, Z., ALLARD, J., OURSELIN, S., COTIN, S., AND PASSENGER, J. 2008. Efficient nonlinear FEM for soft tissue modelling and its GPU implementation. In *ISBMS*, 28–39.

KRÜGER, J., AND WESTERMANN, R. 2005. A GPU framework for solving systems of linear equations. In *GPU Gems 2*. Addison-Wesley, ch. 44, 703–718.

MENDOZA, C., AND GARCIA, M. 2010. Soft bodies using finite elements. In *Game Physics Pearls*. A.K. Peters, ch. 10, 217–250.

TONGE, R., WYATT, B., AND NICHOLSON, N. 2010. PhysX GPU rigid bodies in Batman: Arkham Asylum. In *Game Programming Gems 8*. Cengage, ch. 7, 590–601.

ZHANG, Y., SOLENTHALER, B., AND PAJAROLA, R. 2008. Adaptive sampling and rendering of fluids on the GPU. In *Proc. of Symp. on Point-Based Graph.*, 137–146.