Physically Based Simulation Using Particle Systems

Yalmar Ponce Atencio Claudio Esperança
Federal University of Rio de Janeiro
Computer Graphics Laboratory
{yalmar, esperanc}@lcg.ufrj.br

Abstract

This work consists of extending the rigid body simulation method presented by Harada [3]. Firstly, the Newtonian formulation is replaced by impulse based physics [2]. Secondly, deformable body simulation is achieved by using two different approaches: non-zero volume objects are handled by an adapted meshless shape matching technique[5], whereas objects such as cloths and ropes are simulated using Jakobsen's approach [4].

1. Introduction

Physically based simulation of rigid and deformable bodies has been intensively researched during the last decade. The idea is to obtain a fast and realistic, although not (in general) physically correct, animation. Its main area of application is in entertainment, such as animation films or computer games.

In this work we propose a unified physically based simulation technique, which supports many kinds of objects: rigid, deformable with volume, cloths and ropes. Since all objects are represented by particle systems, they share the same collision detection engine. Further, collision response is computed by integrating collided particle reaction forces for inferred displacements.

2. Collision detection

As described in [3], objects are represented by particle sets, where each particle is a sphere with small diameter. Thus, the collision detection process essentially detects collisions between small spheres. Complex objects can be handled by employing a voxelization method in order to place particles onto the object boundary (see the figure 1).

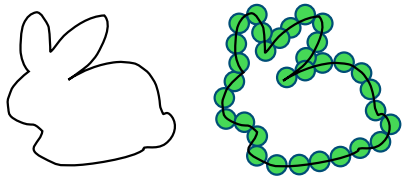


Figure 1. *Discretizing the boundary of a complex object.*

3. Collision response

Once all colliding particles are determined, Harada estimates the force F_B and torque τ_B that will be applied to a given body B by adding up all contributions computed for its associated particles, i.e.,

$$F_B = \sum_{i \in B} f_i \quad \tau_B = \sum_{i \in B} r_i \times f_i,$$

where r_i and f_i are the relative position and the force of particle i respectively. In contrast, we employ a slightly different formulation, using impulses for collided particles associated with rigid bodies. Applying an impulse at a collided point (the midpoint between two collided particles) changes the linear and angular velocities of the body according to

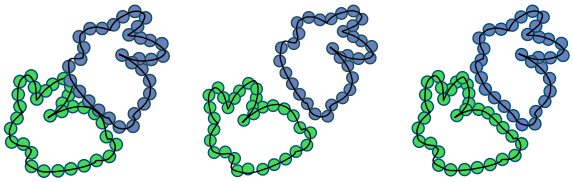
$$v' = v + J/m$$
$$\omega' = \omega + I^{-1}(r \times J).$$

For a particle associated with a deformable object, rather than submit it to a force or impulse, the particle is merely pushed to a non-colliding state.

4. Integration

Once the new states of colliding particles are computed, a new state for the object to which they belong must be estimated. This is done by integrating all particle contributions. For rigid bodies, the integration follows the approach

of Guendelman et al. [2]. In the case of deformable objects with volume, we use the meshless shape matching technique presented by Mueller et al. [5], which uses a semi-implicit integration method (Figure 2 illustrates collision response for both rigid and deformable bodies). Cloths and ropes are simulated using the Jakobsen method [4], which uses a Verlet integrator supported by a constraint relaxation scheme.



(a) (b) (c)

Figure 2. *A collided state (a) and the state after collision response for rigid (b) and deformable bodies (c).*

5. Preliminary results

Our prototype is implemented using C++ and OpenGL/GLSL. Some results are shown in the figures 3, 4 and 5. In the examples, we use deformable cubes with 386 particles, bunnies with 402 particles, cloths with 625 particles and each domino piece has 15 particles. All the examples run at interactive rates.

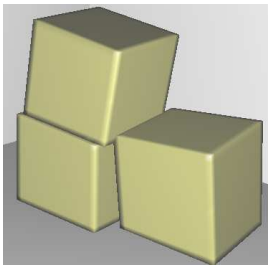


Figure 3. *Three deformable cubes in contact.*

6. Conclusions and future work

As shown by Harada, the method can be executed entirely in GPU by storing particle data in textures. Since modern GPUs support 4096x4096 textures, thousands of objects can be simulated. Since our current prototype is still CPU programmed, our next step is to reimplement it in GPU using the CUDA [6] technology..

The particle-based method works well for small time steps. On the other hand, fast moving particles or large time steps may lead to an ill-conditioned simulation. We are currently at work trying to improve the collision detection process in order to support larger time steps and objects with different particle sizes by adapting the technique presented by Le Grand [1].

Notice that although objects are represented by coarsely geometry for quick collision detection, rendering can be performed using more detailed representations. Our prototype, at this time, does not support interaction with fluids yet, but the extension is natural.

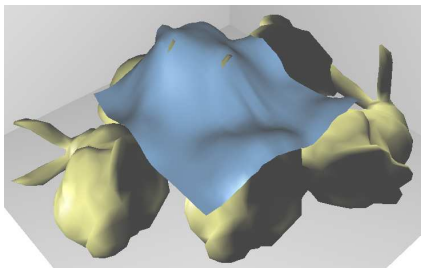


Figure 4. *A Cloth falling on top of bunnies.*

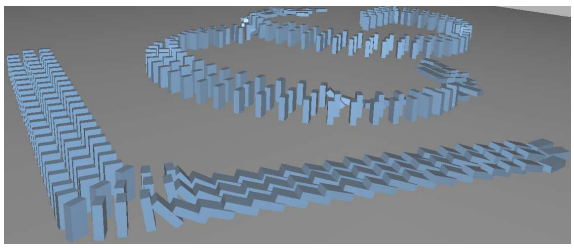


Figure 5. *Simulating a domino configuration.*

References

[1] S. L. Grand. *GPU Gems 3*, chapter Broad-Phase Collision Detection with CUDA, pages 697–721. Addison Wesley, 2007.

[2] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 871–878, New York, NY, USA, 2003. ACM.

[3] T. Harada. *GPU Gems 3*, chapter Rigid Bodies on GPU, pages 611–632. Addison Wesley, 2007.

[4] T. Jakobsen. Advanced character physics. In *Proceedings, Game Developer's Conference 2001*, SJ, USA, 2001. GDC Press.

[5] M. Muller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. In *Proceedings of SIGGRAPH'05*, pages 471–478, New York, NY, USA, 2005.

[6] NVIDIA™. CUDA Environment – Compute Unified Device Architecture, 2007. http://www.nvidia.com/object/cuda_home.html.