Particle System Collision Detection using Graphics Hardware

Dave Knott
University of British Columbia
and *Radical Entertainment*

Kees van den Doel
University of British Columbia

Dinesh K. Pai
University of British Columbia
and *Rutgers, The State University of New Jersey*

1 Introduction

We demonstrate a technique for performing collision detection between dynamically simulated particle primitives and groups of implicitly defined objects. We describe how this collision detection can be performed using a programmable vertex engine.

We use the particle collision detection in the context of visual simulation, which requires no CPU intervention. We also provide a method for reporting collision information back to the CPU for further processing, using a construct called the *impact map*.

2 Particle Simulation

We simulate particles[Reeves 1983] whose behaviour is described by closed-form equations of motion. A particle's defining parameters are specified once and never change. This allows us to create simulations in which the properties of particles are computed on graphics hardware using vertex programs/shaders[Lindholm et al. 2001]. For purely graphical applications, the computer's CPU need never participate, except to pass data to the graphics accelerator at each frame of the animation.

The motion of particles is completely determined by initial conditions, with the exception of impacts with (and possible response to) interfering bodies. Instantaneous particle position is given by a parametric equation $\mathbf{p} = g(t)$, where t is global simulation time. Particles may impact with *colliders*, surfaces defined implicitly by some equation, $f(\mathbf{p}) = 0$, where $\mathbf{p}$ is a point on the surface.

The intersection of a particle's motion with the surface is then described by $h(t) = f(g(t)) = 0$, which implicitly gives the collision time, t. Each collider may also have associated constraints, $c(\mathbf{p}) > 0$. These are useful in detecting collisions with bounded planar sections, or objects with holes, for example.

We can currently test for collisions between planar sections and particles obeying second-order dynamics. We can also test for collisions between first-order particles and quadric surfaces.

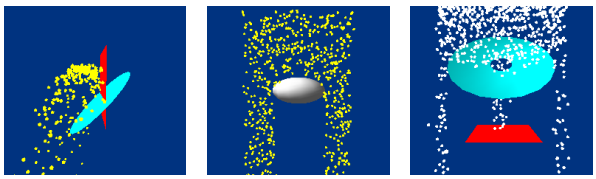

Figure 1: Examples of collision

3 Collision Feedback

The *impact map* is a two-dimensional representation of the locations on colliders at which particles have impacted. At each simulation step, a GPU vertex program computes whether or not each particle has impacted a collider at some time between the previous and current frames of the animation. If a collision is detected, then a two-dimensional representation of the exact impact location is calculated. A point primitive is drawn into the impact map at that location. The colour channels of the impact map can be used to store information about collisions, such as impact energy.

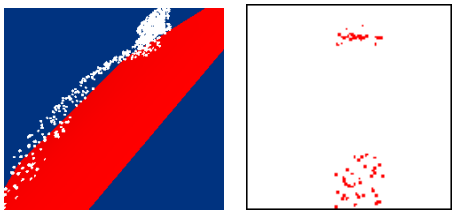

Figure 2: Collision with response and the associated impact map

Applications such as audio simulation require collision information at rates much higher than those achievable with graphics hardware. The closed-form nature of the simulation allows us to achieve super-resolution results. Exact collision time is reported in the impact map, which is rendered slightly ahead of the visual simulation.

For static or precomputable collision information, we can save CPU cycles by storing the data in texture memory. Collision data is reported in the impact map via a dependent texture look-up.

4 Results

We have implemented an audio-visual simulation of hail falling on an outdoor scene. Hail is simulated as particles with motion defined by second-order dynamics. We perform collision detection between each hail particle and a variety of objects, as shown in Figure 3.

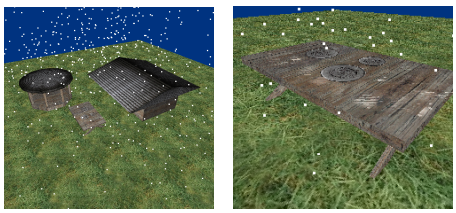

Figure 3: Hailstone particles colliding with outdoor objects

We use collision feedback for two purposes. For visual simulation, a "decal" texture is rendered wherever a hailstone impacts with a collider object in the scene. We also drive an audio simulation that uses modal synthesis to generate sound[van den Doel et al. 2001] when hail strikes colliders. Future directions include using the dependent texture look-up to report sound synthesis parameters via the impact map.

References

LINDHOLM, E., KILGARD, M. J., AND MORETON, H. 2001. A user-programmable vertex engine. In *Proc. of SIGGRAPH 2001*, 149–158.

REEVES, W. T. 1983. Particle systems - a technique for modeling a class of fuzzy objects. In *Proc. of SIGGRAPH 83*, 359–376.

VAN DEN DOEL, K., KRY, P. G., AND PAI, D. K. 2001. FoleyAutomatic: Physically-based sound effects for interactive simulation and animation. In *Proc. of SIGGRAPH 2001*, 537–544.

contact e-mail: {knott|kvdoel|pai}@cs.ubc.ca