CSCE 645 Research Project Final Report
“Interesting Interactions with Granular Material”

Gabriel Stella

1 Introduction

This semester, I studied the simulation of granular materials using a *Lagrangian (particle-centric)* approach. This task has important applications to industry (e.g., medicinal powders, gravel processing), geology (e.g., landslide modeling), graphics (e.g., for videogames and movies) and more. However, it is notoriously difficult to simulate granular matter both accurately and efficiency due to its particular properties. Clearly, a collection of grains cannot be modeled as a single rigid body, as the geometry and topology of the group can change drastically even under common conditions. While it may be tempting to the collection as a fluid, since grains can flow and even splash, this approach neglects highly important inter-granule forces like shear friction, which are essential in producing macroscopic phenomena such as pile formation.

Approaches to the simulation of granular matter generally fall into one of three categories: *Eulerian* grid-based, Lagrangian particle-based, or hybrid. Grid-based methods follow the lineage of fluid solvers, breaking space into cubic cells and numerically solving equations of motion. Notably, the equations of motion are typically based on the behavior of classical fluids, so they do not accurately model the movement of sand. For example, while the density of a fluid is generally constant, the density of granular materials is not: when in motion, the individual grains are farther apart, and then they settle into a more compact distribution when static. Particle-based methods, while in some sense more "true to reality", also have significant drawbacks; most obviously, the required computation scales up linearly with the number of particles. This is a significant problem, given that a single spoonful of sand can have thousands of individual grains. Because of this, real-time simulation of high-particle-resolution scenes is intractable. Hybrid methods attempt to merge the benefits of the two approaches, relying on a grid-based approach for many of the calculations and using a reduced number of particles to ensure accurate advection, but they introduce new problems, such as energy dissipation (due to representation transfer between particle- and grid-type) and instability.

2 Implementation

To keep my system simple and intuitive, I took the particle approach. My first implementation followed the work of [1], using the *molecular dynamic* (MD) approach. This technique is based entirely on the calculation of contact forces (i.e., particle-particle repulsion and friction). Unlike typical rigid-body methods, it allows for slight overlap between particles in order to determine the strength of these forces. While this is theoretically acceptable, since the contact forces should prevent significant interpenetration, in reality some problems arise. As discussed in [2], force-based constraints are numerically intractable; in this case, either restitution is low, and overlap is allowed, or restitution is high, and any nonnegligible overlap results in what is essentially an explosion (so, you better hope that your constraints aren’t violated). In my experiments, as a particle pile got taller, pressure seemed to indefinitely increase in the bottom particles, leading to overlap (in some cases, enough overlap to exit the static particle barrier). This is contrary to what is described in [1], which specifically claims that pressure becomes constant beyond a certain depth. I believe this discrepancy has at least two causes: first, they use significantly smaller timesteps, since they do not focus on real-time interaction; second, this behavior is significantly reduced when using polysphere (or polycircle) particle ensembles as opposed to single-sphere (or single-circle) particles.

As a comparison, I also implemented a proper rigid-body circle simulator. This second simulator enforces noninterpenetration by adjusting particle positions and velocities upon contact, while also ensuring conservation of momentum and energy (energy can be dissipated if restitution is lowered). This simulator is both more efficient and more qualitatively accurate (by my subjective judgment; it just *feels* a lot more like the sand I'm used to). While the MD implementation allows overlap and experiences oscillations when forces are delivered into a collection of particles (e.g., via impact), akin to a fluid being disturbed, the rigid-body approach displays proper particulate splashing and rapid settling. Unfortunately, it is rather arduous to extend this technique to support polycircles; short of constructing a fully-fledged physics engine with an iterative constraint solver, there is no good solution. On the other hand, the MD approach is rather trivial to extend to polycircles.

In light of this, I experimented with several approaches to try to reconcile the two techniques. Inspired by the description of the constraint solver in [2], I first attempted to modify the MD approach to prevent interpenetration. For each particle, my method calculated the "safest direction" (a vector pointed away from its colliding neighbors, weighted by overlap) fit either the current force or current velocity to that direction, with varied results. If the force was fit to this direction, overlap was slightly reduced, but significant oscillations were introduced to the system as the particles in a pile would move back and forth in an attempt to escape their neighbors[1]. If the velocity was fit to this direction, overlap was prevented and there were no oscillations; however, conservation of momentum would be violated. This was most noticeable when dropping a large group of particles; as they fell to the floor, midair collisions would often result in one particle suddenly stopping so as to not enter the other. Ultimately, it seemed that the only solution to enforce a good amount of rigidity would be to convert the MD approach fully to a rigid body approach.

Fortunately, the polycircle MD implementation suffers far less from the problem of overlapping. Still, since I was so mesmerized by the behavior of the rigid circle simulator, I wanted to attempt to extend it to polycircles. Whereas MD circles are rigidly bound into 3-bodies, I decided to bind my rigid circles using springy attractive forces. While this approach is definitely not without its drawbacks, it produced such nice piles that I decided to keep it.

3 Results

In short, I have implemented 4 different 2D sand simulators: MD circles, MD polycircles, rigid circles, and rigid polycircles. In my opinion, my primary contribution is these implementations, especially since they are open-source (they are browser-based, implemented in Javascript, so any user can easily view the source; they are also on GitHub). Thanks in addition to the fact that they run in real time and have several configuration options (e.g. force visualizations, real-time-adjustable physical parameters) and multiple tools with which the user can interact with the simulation (e.g., create, destroy, or move particles), I believe these simple implementations can significantly lower the barrier to those interested in experimenting with physics simulations. They would also serve as great jumping-off points for reimplementation in another language, such as C++; this would drastically increase the number of particles that could be simulated. Finally, with the current design, it would be quite easy to extend the simulations, e.g. by incorporating fluids that interact with the sand particles.

The MD simulators follow exactly from the description in [1]. The single-circle particle simulator, though it suffers from the interpenetration problem discussed earlier, is a fascinating demonstration of the way that simple rules (i.e., the normal and tangential forces between particles) can give rise to incredible

[1]It did make for a very amusing jell-o simulator, though.

complexity. For example, if a particle resting atop a flat surface of other particles begins to spin, the tangential friction forces will cause it to begin moving along the surface; similarly, if a nonrotating particle is dragged along a surface, friction will cause it to rotate. Alternatively, if a particle impacts a cluster of other particles, the normal forces naturally give rise to a wave of energy that dissipates through the cluster, while a subsurface wave can cause particles to splash out from the surface. In this way, the system does seem to be a better model of e.g. water than of sand; sand doesn't usually jiggle when impacted, or at least not on the beaches I've been to.

The polycircle MD implementation fairs slightly better, displaying reduced overlapping and the ability to support a slight angle of repose; still, the particles gradually settle into a relatively flat pile. On the other hand, the rigid polycircles do exhibit a significant ability to interlock, as promised in [1]. In addition, they are an excellent system for demonstrating one of the most interesting properties of granular materials: *anisotropic force chains*. Due to the uneven arrangement of granules within a cluster and their asymmetric shapes, forces are not transmitted uniformly through the medium; instead, long chains of increased directional forces are present in the material. In the simulation, as a pile gets larger, the forces at the bottom become clearly visible and nonuniform, forming obvious chains.

The rigid single-circle system is probably my favorite of the group. Just like with the MD implementation, it is fascinating to see the behaviors that can arise from simple equations; in this case, the preservation of tangent velocity and exchange of normal velocity (i.e., circles reflect upon collision) leads to incredibly real-looking reactions. For example, simply dragging one of the circles from the starting configuration (10 randomly-positioned circles in zero-gravity) will cause a wonderful cascade of collisions and movement. This system is in some ways even simpler than the MD approach; in addition to being computationally more efficient, it has only a single parameter, restitution (it could also have a friction parameter, which would decrease tangent velocity upon collision, but I decided against it for now), while the MD approach has 5 parameters, whose effects on the simulation are often unclear.

In addition to supporting important properties of granular material, like realistic impact splashing and a significant angle of repose, the rigid circle system also displays a phenomenon I find particularly interesting: the formation of "crystals". As particles fall and settle into a pile, they form discrete groups; particles within the groups are tightly packed (each with 6 neighbors), with voids running along the edges of each group. Just as in reality, the size of the groups generally increases with the time taken to settle: longer formation times produce larger crystals. Interestingly, since the particles can move, they sometimes gradually settle further and fill in these cracks (it is especially fun when this happens due to an impact).

Finally, my experimental approach was to join rigid circles in groups of 3 using springy attractive forces. This causes some strange behaviors, such as particles experiencing a net force during freefall and deviating from a straight path downwards (usually accompanied by spontaneous angular acceleration). While this is very much a prototype, I decided to keep it available, since it produces good piles and may be interesting to experiment with.

Each of the described implementations is available to play with in-browser on my website, on the "Toys" page.

- "Circles" implements single-circle MD
- "Billiards" implements single-circle rigid bodies
- "Soft Sand" implements polycircle MD
- "Hard Sand" implements the rigid-body polycircles

Each comes with a large set of configuration options, both visual and behavioral, as well as several tools to interact with the simulation in real time.

4 Future Work

I was unfortunately not able to implement all of the plans I began the semester with. One of the plans, designing a compression algorithm, turned out to be somewhat ill-defined; while skipping inactive particles could save computation time, there is no way to reduce the space complexity without losing information about particle locations, which would open the door to all kinds of additional problems. A second plan was to reimplement the algorithms in C++ once I was satisfied with the web demo; I intentionally decided against this, instead focusing on perfecting the web demo so that people could more easily play with my simulations. A third was to add an additional fluid layer that would interact with the sand particles and allow for the construction of more complex shapes (e.g., self-supporting wet sand arches). I sadly did not get to this, since I spent so much time debugging and perfecting the basic behaviors and interactivity of the simulation, but I believe the current framework would easily admit this sort of extension.

I actually spent quite a significant portion of the project dealing with a very specific bug, which I called "popcorn" because it would cause the entire scene to explode like some sort of movie-theatre popcorn machine. As it turns out, I simply had a tiny error in my calculation of local velocity (i.e., velocity at a point on a rotating circle)... since I was so new to working with physics engines, fixing this problem took a very long time and prevented me from making good progress on the rest of the project.

Regardless, I am incredibly happy with the various implementations I have. They are fairly well-optimized, supporting hundreds (or even thousands) or particles in-browser in real time[2], and I *definitely* got the "interaction" part down: I could play (...and have played... but it counts as work because it's *testing*) with these simulations for hours. Plus, though they are little more than a prototype at this stage, I'm pretty sure my springy rigid-body polycircles are a novel idea, and it's always fun to have something totally your own.

In future work, I think it would be most interesting to pursue the proper rigid-body approach, with a full constraint-solving physics engine (though likely still just supporting circles, or perhaps spheres). This would still admit extensions to include e.g. fluids, which I think would be very interesting, and has slightly better qualitative performance than the MD approach *without* the use of such large contact forces. Finally, it would allow a very common technique to be used to save computation time: putting stationary bodies to "sleep" after some time, and only awakening them upon being affected by some other body. Of course, this would all be even more interesting in a highly-optimized C++ implementation, which could support tens of thousands of particles in real time. I definitely believe that this type of interactive sand simulation is an interesting topic to explore.

References

[1] N. Bell, Y. Yu, and P. Mucha, “Particle-based simulation of granular material,” 01 2005, pp. 77–86.

[2] A. Witkin and D. Baraff, “Physically based modeling,” Available at https://graphics.pixar.com/pbm2001/ (2001).

[2]The system actually executes multiple small timesteps in every frame, meaning e.g. to run at 60 visual fps, it is running 600+ logical fps!