Piles of Objects

Shu-Wei Hsu
Texas A&M University

John Keyser
Texas A&M University
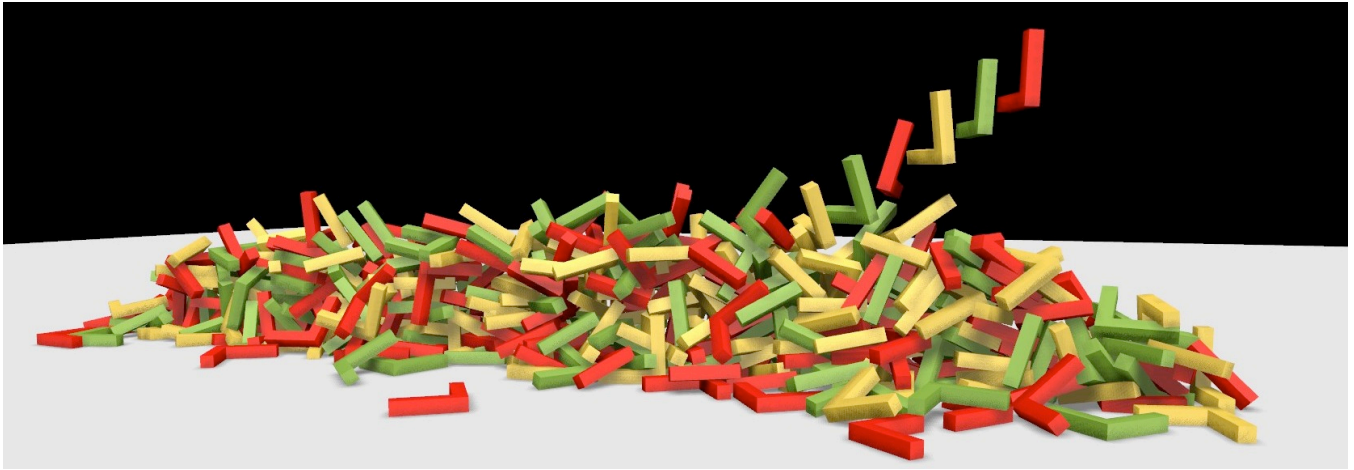
Figure 1: *L-shaped objects piling up. This is an image from Example 5—Drop in a Line (see Section 4).*

Abstract

We present a method for directly modeling piles of objects in multibody simulations. Piles of objects represent some of the more interesting, but also most time-consuming portion of simulation. We propose a method for reducing computation in many of these situations by explicitly modeling the piles that the objects may form into. By modeling pile behavior rather than the behavior of all individual objects, we can achieve realistic results in less time, and without directly modeling the frictional component that leads to desired pile shapes. Our method is simple to implement and can be easily integrated with existing rigid body simulations. We observe notable speedups in several rigid body examples, and generate a wider variety of piled structures than possible with strict impulse-based simulation.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism-Animation; I.6.8 [Simulation and Modeling]: Types of Simulation-Animation

Keywords: rigid body simulation,piling,angle of repose

1 Introduction

In this paper, we introduce a method for representing piles of objects in large multibody simulations. Object piling and stacking presents one of the more interesting phenomena, but also one of the more computationally intensive situations, in such large simulations. For this reason, a method that can handle piling conditions at faster rates with more user control would be desirable. This paper aims to provide such a method.

There are two major reasons for implementing a specific piling approach. First, by representing piles of objects, we can deactivate objects within the pile, saving significant time. This offers improvement over standard methods for deactivating objects, based strictly on object momentum.

Second, and perhaps more important, stacking behavior that one may wish to see is often influenced by friction among the simulated objects. Simulating static and dynamic friction accurately is a much more complex process than simple impulse-based collision response, and is thus avoided in many simulations. However, by avoiding friction, one is limited in the types of stacking and the shapes of piles that can be achieved. Our piling method allows us to specify the shapes of piles that can be obtained, allowing for a more realistic or more art-directed look to emerge.

The main result presented in this paper is a model for piles of objects in which layers of objects are explicitly represented. We present a method for determining where these layers should be and updating the layers as objects pile up. The only change to the fundamental simulation is a new criterion, based on the pile structure, for marking objects "asleep" and stopping their simulation. This not only enables us to improve computation time, but also lets us provide greater user control in the structure of the piling shape.

2 Background

There is a wide body of work on collision detection, collision response, friction, etc. Some of the foundational work includes that of Baraff [1989; 1994], Hahn [1988], and Mirtich and Canny [1995]. A comprehensive review of this material is beyond the scope of this paper, but summaries of rigid body simulation techniques can be found in a variety of books and websites (e.g. [Ericson 2005]).

Within these methods, contact forces have been modeled a variety of ways, ranging from analytical methods [Baraff 1994] to methods based on series of collision impulses [Mirtich and Canny 1995]. Recent work has focused particularly on modeling of contact forces for both rigid and deformable objects [Kaufman et al. 2008; Otaduy et al. 2009]. However, solving for all the forces in a large stack or pile is an NP-hard problem [Drumwright 2008]. To get around the problems of modeling all inter-object forces, several models for more directly dealing with stacking and contact have been proposed in recent years [Guendelman et al. 2003; Erleben 2007; Drumwright 2008]. Included in these approaches are techniques that, like our technique, try to stop or temporally fix some objects within a random pile to increase efficiency [Guendelman et al. 2003; Kaufman et al. 2005; Schmidl and Milenkovic 2004]. Other methods have dealt more directly with the piles that can result from more structured stacking [Erleben 2007].

Rather than trying to model the individual contact and stacking forces and have a pile result, our approach aims to model the pile itself and make the objects adapt to it. Compared to papers on stacking, contact, and friction, papers dealing explicitly with piles are much more limited, especially in graphics. As objects fall, a conical hill tends to be formed little by little. An interesting fact is that no matter how many of the same objects are piled up, there exists a maximal stable slope of that pile. The angle of the slope is called the angle of repose [Zhou et al. 2002; Liu and Zhou 2008]. This concept has been used previously in graphics to generate realistic models of items (such as snowflakes) in a final position [Fearing 2000], but has not been used directly in simulation until now.

The maximal angle of repose is related to several factors: the size of objects, the shape of objects, and the material friction. In order to generate piles "correctly" within a simulation, simulation of the contact frictional forces is necessary. This tends to be a much more complex and time-consuming approach, and so often the models of static and dynamic friction are replaced with simplifications, e.g. frictional components within an impulse response. The net result is that while a typical simulation will support some piling due to object geometry and the limited approximations of friction, the piles that result are often shallower (i.e. a lower angle of repose) than those that might be desired.

3 Generating Piles

To generate piles, we create a spatial structure that we refer to as the "stacking structure." This structure will be used to explicitly represent the pile(s) of objects. Objects that are determined to be part of the pile are put to "sleep," i.e. they are no longer simulated, unless later reactivated.

3.1 Stacking Structure

Our stacking structure is a spatial data structure that is used to track the density with which objects are packed in a pile. A cone shape is used to track the stacking activity. The stacking structure location may be specified manually (when a particular pile location is known or desired), or set based on where small sets of objects initially land. This cone shape is defined by an angle of repose, α, that is the one user-defined parameter. α can be determined by measuring the shape of a pile of objects in an precomputed simulation, or more typically may be set by the user to a desired value. In this way, α can reflect the contributions of frictional components that are not directly simulated.

Consider a simulation of N objects of a single type. The volume of each object is v_{obj}, where we use a conservative bounding volume to approximate volume. A conservative measure for v_{obj} is

preferable to an exact volume, since the volume measure should reflect the amount of space the object occupies in a pile, rather than the true object volume. We typically use a bounding box for this estimate, but observe no significant difference in behavior of the simulation when similar estimates are used. We also determine the maximum length of the object in any direction, D, which we can approximate by a bounding box diagonal. We assume those objects will pack together to form the cone, making the volume of the cone approximately Nv_{obj}. Given an angle α for the cone, we have a relation between the base radius r and height h such that $\tan \alpha = h/r$. We can thus determine the base radius from:

$$Nv_{obj} = \frac{\Pi}{3} r^3 \tan \alpha \qquad (1)$$

We generate a spatial representation of this structure bottom-up. We cut the cone into k layers, where $k = \lfloor h/D \rfloor$, ensuring that no object can span more than two layers. For each layer i there is a corresponding radius r_i and volume v_i, obtained by simple geometry. The maximum capacity c_i of layer i is set to v_i/v_{obj}. Thus, a stacking structure is defined by $(\alpha, k, r_1 \ldots r_k, c_1 \ldots c_k)$, although many of these values can be computed, rather than stored.

3.2 Growing the Pile

During simulation, more and more objects will fall into the stacking structure, causing it to grow. We use an occupation ratio to determine whether the structure needs an update or not. Let m_i be the number of objects in layer i of this structure. To collect m_i, we check the location of the center of mass of each object. We also compute the density of the layer, $\rho_i = m_i/c_i$. These values (m_i and ρ_i) are maintained continuously throughout the simulation.

The occupation ratio ρ can be computed as $\sum(m_i)/\sum(c_i)$. If $\rho > 0.5$, we will update the structure. To update the structure, we simply regenerate the structure again with $N = 2\sum(c_i)$ (i.e. approximately doubling N); the angle is still α.

The center of a pile can be updated over time by gradually shifting the x, y coordinates of its center to reflect the average of those objects in the pile.

3.3 Causing Objects to Sleep

As in other simulation approaches, we achieve efficiency by deactivating objects that are either hidden from view or not moving. We refer to this as putting the objects to "sleep." Determining whether an object should go to sleep relies on three things: how dense the layer the object belongs to is, how close the object is to the center of the layer, and how slowly the object is moving. If an object is in a dense layer, close to the center, and slow enough, we can aggressively force it to go to sleep.

Assume an object is located on layer i. We use $\rho_i = m_i/c_i$ as the first measurement, which is the density of layer i.

To measure closeness to the center, we use the following equation:

$$d = \max(r_i - \sqrt{p.x * p.x + p.y * p.y}, 0)/r_i \qquad (2)$$

where p is the position of the object's center of mass, with z being vertical (we assume the pile is transformed to lie at the origin). If the object is located on the center of the layer, the value will be one; if the object is not inside the cone, the value will be zero.

The last measurement, speed, is computed as $v = 1/(1 + |v|)$, where $|v|$ is the magnitude of the object's linear velocity. v ranges from 0 (at infinitely high velocity) to 1.

To determine when an object goes to sleep, we multiply these three measurements together (i.e. $\rho_i dv$) to get a sleep score. If the score is greater than some threshold, τ, the object is ready to be moved to a sleep state. This score will tend to make objects go to sleep from the center of the pile toward the outside. However, we also want to ensure that objects sleep bottom-up (so that objects on top of the pile can move if those underneath are still moving). To do so, we check whether all levels $k < i$ contain sleeping objects (i.e. are dense and contain objects close to the center). If so, we allow the object to be put to sleep.

3.4 Avalanching and Waking

The procedure outlined above is sufficient to allow objects to pile up, and stack nicely in ideal situations. However, there is nothing in the above that would prevent a small pile to form, objects to be put to sleep, and then more objects pile on top, creating unbalanced and unrealistic piles. There needs to be a way to "wake" objects, as well as to detect conditions in which the pile shape is not being maintained. We use an avalanching technique to detect such cases.

Avalanches form when the upper levels of a pile cannot be supported by those underneath. This will happen when a pile temporarily exceeds its natural angle of repose. Note that this can occur at any point along the pile; it is a local effect on the pile, not a global one. In our pile structure, this is detected by examining the density of each layer of the pile. We trigger an avalanche whenever we detect that a given layer has density greater than some threshold (we use 0.8 in our implementations). When an avalanche is triggered, we temporarily wake up all objects, and allow them to undergo motion. When objects are awoken, they will simulate for several timesteps before they are checked against the sleeping threshold. This allows them to sufficiently resolve any forces or imbalances. Most objects not in the unbalanced region of the avalanche will quickly return to a sleeping state, but those in the unbalanced area will fall to lower areas. The result is an effect that one expects to see, where piles that become unbalanced have regions that slide down to more stable configurations.

Objects in the pile can also be woken when an external force (such as from a high-momentum colliding object) hits the pile. Such forces may propagate throughout the entire pile in a single time step, and thus it is necessary to wake the entire pile when such an event occurs. We use a momentum heuristic for waking in such cases—i.e. if an object with sufficiently high momentum hits a pile, the pile is woken so that the force propagation can be resolved.

3.5 Pile Boundaries and Overlapping

Pile growth can be constrained in various ways. If a scene contains fixed boundaries, the pile growth can be constrained so that no layers may extend past the boundaries (see Example 6 in Section 4.1). If the lowest layer reaches full density, but is touching the boundary, it is not expanded further. Also, a constrained pile's center can be moved vertically. This allows modeling of piles of objects within containers, where the pile will build in the container, fill it up (creating a rough cone over the container), and eventually spill over the sides.

Multiple piles can easily be used within a single simulation (see Examples 3 and 5 in Section 4.1). Note that single objects might contribute to the density measurement of more than one pile. Also, two piles whose centers lie near enough to each other and are of the same type can be combined into a single pile.
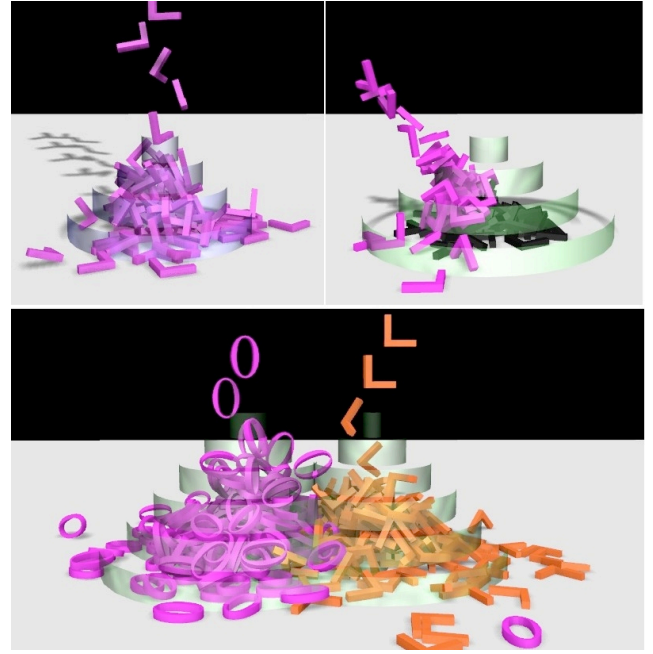

Figure 2: *a. Upper left: Example 1—Drop in a Pile. b. Upper right: Example 2—Varying Drops. c. Bottom: Example 3—Multiple Piles. The semi-transparent cylinders show the layers of the stacking structure*

4 Implementation and Results

We have implemented our piling approach within a rigid body simulation. We use Swift++ [Ehmann and Lin 2001] for collision detection, and a basic impulse-based collision response. All results are collected on a system with a Core 2 Duo 2.66 processor, 4 GB of RAM, and an NVidia 9800GT card.

4.1 Examples

To explore the usability of our piling approach, we use six different examples. Examples 1 through 4 use piles located where the first object hits, while Examples 5 and 6 use manually specified piles. The accompanying video shows animations, and Table 1 gives details of the settings and results of the examples.

Example 1, Drop in a Pile (see Figure 2a): In this example, we drop several L-shaped objects onto a plane. Objects are dropped one-by-one directly down. This example demonstrates how the structure grows, how the objects stop, and how avalanches happen. In the pictures/videos, we use transparent cylinders to show the layers of our pile structure. Sleeping objects are colored black. The result shows that sleeping objects generally build inside-out and bottom-up. Also, when the pile becomes too high, all objects wake up and the simulation undergoes an avalanche effect.

Example 2, Varying Drops (see Figure 2b): In this example, we drop objects from different directions, and vary the timing over which they drop. We start by firing one string of objects from a non-perpendicular angle to a plane. When the objects have piled up, we fire another string of objects from a different angle to that pile again. The result shows that the structure can easily deal with discontinuous dropping.

Example 3, Multiple Piles (see Figure 2c): Another characteristic of our stacking structure is that it is able to use multiple stacking
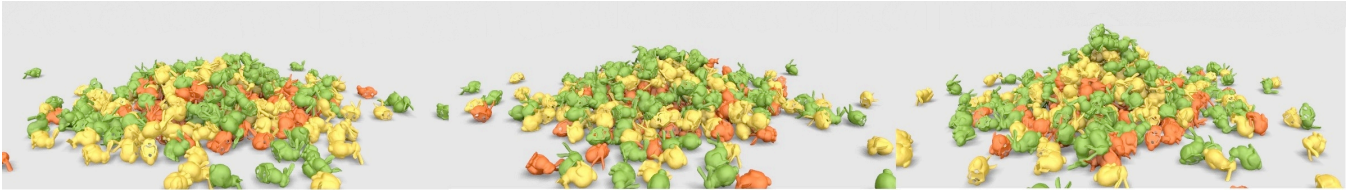
Figure 3: *Example 4—Bunny Matrices. The results from three different runs, with three different angles of repose, are shown.*
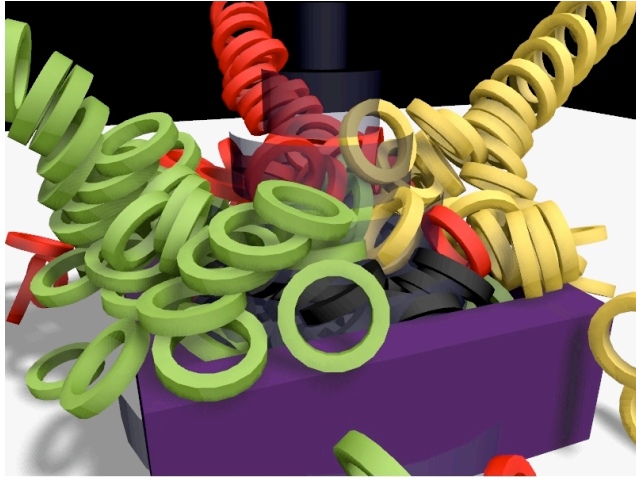

Figure 4: *Example 6—Rings in a Box.*

Figure 5: *Timing and Error for Example 5—Drop in a Line. Left axis shows time per frame (in seconds), and right axis shows relative error. Scaled absolute error is superimposed.*

structures and to mix them together. In this example, L-Shape and O-Shape objects are used. Again, they are dropped straight, parallel, and close to a plane. We generate two stacking structures. The result shows that although objects in the two piles are put to sleep separately, mixing them together still creates a reasonable sleeping distribution, and piling behaves correctly.

Example 4, Bunny Matrices (see Figure 3): We drop a set of bunnies organized in a 3D matrix, in order to quickly form a pile. We use this example to demonstrate varying angle of repose in the piles. We run the simulation three times, using three different angles of repose. The smallest angle matches the angle of repose calculated from a prior run of the full simulation, and the other two use larger angles. The difference in angle is clearly seen, and all piles appear plausible. Our structure handles this situation, however the performance improvement is limited, compared to other piling examples. The density of the top layer is always high when these matrices are falling down, therefore the structure cannot allow objects to sleep while objects remain at or above the top layer.

Example 5, Drop in a Line (see Figure 1): We drop a set of L-shaped blocks, where the drop point gradually moves back and forth along a line. Seven pile structures are used, forming a line. We use this example to demonstrate that piling can be used to create non-circular piles. That is, even though the individual piles might be generated from cylindrical stacks, the combination of piles does not have to yield something like a single pile.

Example 6, Rings in a Box (see Figure 4): Dropping objects into a container is one example of a possible situation for our simplified stacking method. We prepare a box and set a stacking structure in the center of this box. As objects pile up, the objects on the lower layers become occluded, making it safe to put them to sleep. In addition, we want objects to sleep only inside the box. To do so, once the size of the lowest layer exceeds the boundary of the box,
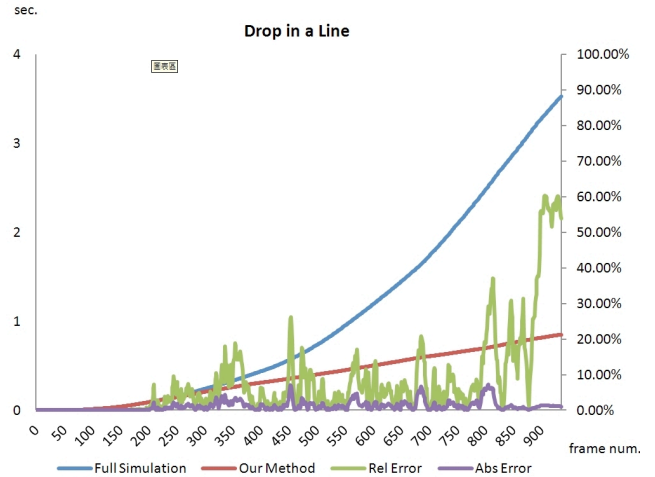
the structure will stop growing.

Other Examples (see video): The accompanying video contains a wider range of examples than we can discuss here. This includes a larger pile structure (10,000 objects), multiple sized objects, and a pile being broken up after being hit by a high momentum object.

4.2 Performance and Error

The overall performance of the example scenes is summarized in Table 1. In addition, we show charts with the timing and error of Examples 5 and 6 in Figures 5 and 6, respectively. We discuss here some details of the results.

4.2.1 Behavior

Visual examination of the various examples verifies that we are able to achieve the primary behavior we desired, namely the forming of piles of objects. Figure 3 illustrates different pile angles achieved from the same basic simulation. The simple impulse-based simulations that we use cannot, on their own, achieve the higher angle of repose that might often be desired, without resorting to much more expensive calculations of friction. Comparison to the full simulation without piling (see examples on video) demonstrates that we are able to achieve desired stacking behaviors (i.e. taller stacks) that are not achieved by the basic simulation alone.

4.2.2 Timing

As can be seen in Table 1, there is a speedup from using the piling method, in all simulations. Figures 5 and 6 illustrate that this

	Example	Angle	# Frames	# Objects	Full Sim. (sec)	Piling (sec)	Speed-Up	Error (%)
1	Drop in a Pile	45	1818	400	4125.4	681.5	6.0x	5.8
2	Varying Drops	30	856	140	495.2	231.9	2.1x	20.8
3	Multiple Piles	45	2203	400	5571.2	1488.2	3.7x	5.1
4a	Bunny Matrix I	20	312	300	596.1	502.3	1.2x	9.3
4b	Bunny Matrix II	35	312	300	596.1	359.4	1.7x	16.0
4c	Bunny Matrix III	45	312	300	596.1	333.4	1.8x	4.2
5	Drop in a Line	45	928	400	1461.6	600.8	2.4x	9.6
6	Rings in a Box	60	1061	450	4904.2	2557.2	1.9x	9.9

Table 1: *Overall timing results for examples. Angle is the angle of repose used in the example. # Frames and # Objects are the number of frames for the full simulation, and the number of objects used in it, respectively. The two timings are for the "full" simulation (without piling) vs. the simulation with piling, and Speed-Up gives the relative difference over the course of the entire simulation. The error computes the average difference in movement relative to the full simulation (relative error discussed in Section 4.2.4), over the course of the entire simulation.*
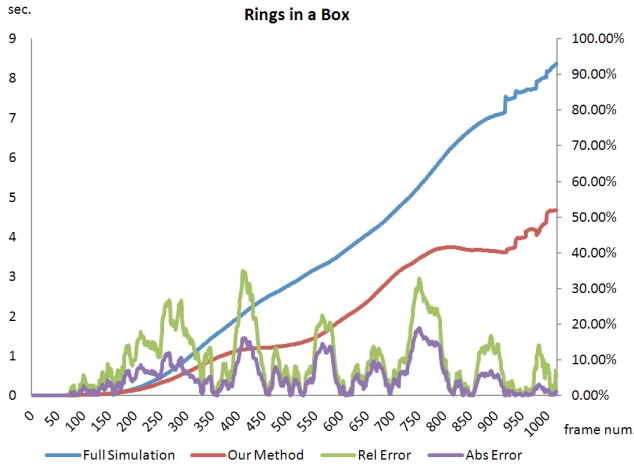

Figure 6: *Timing and Error for Example 6—Rings in a Box. Left axis shows time per frame (in seconds), and right axis shows relative error. Scaled absolute error is superimposed.*

τ	none	0.7	0.6	0.5	0.4	0.3
Time(s)	495.2	412.3	269.8	231.9	215.6	199.4
Speed-Up	1.0x	1.2x	1.8x	2.1x	2.3x	2.5x

Table 2: *Effect of sleeping threshold, τ, on performance of Example 2. Lower τ values give greater performance improvement, but very low values can cause unnatural-looking piles.*

speedup on this example. This is comparable to, but somewhat lower than the speedups we see in our piling approach. Thus, we claim that our piling approach offers a moderate improvement in performance, compared to simple sleeping based on velocity.

4.2.3 Sleeping Threshold

The threshold, τ, used to determine when an object is put to sleep (see Section 3.3) has an impact on both the behavior and efficiency of the simulation. Table 2 shows timing results for Example 2, with varying values of τ, and the accompanying video shows a visual comparison. Lower values of τ, indicating more aggressive sleeping, do yield a greater performance benefit. However, for very low values of τ, piles may begin to look unnatural, as some objects are put to sleep before they are truly at rest. Very high values of τ may result in little piling action at all. In all examples in this paper, we use $\tau = 0.5$, which provides a good performance improvement, while generating believable pile structures.

4.2.4 Error Measure

Measuring error in this sort of simulation is not straightforward. In particular, the ability to set an angle of repose, while producing more useful simulations, also creates "error" vs. a basic implementation that has a de-facto (and likely different) angle of repose. The examples we have tested (with $\tau = 0.5$) all look plausible, however this is a difficult feature to quantify.

In order to have some quantitative measure of the error our method introduces, we have developed an error metric based on relative motion of individual objects. Assume x_i^j is the position of (the center of) object i at frame j in the full simulation (without piling), and let $\tilde{x}_i^j$ denote the position in the simulation with piling. We calculate the frame-to-frame motion of the objects: $m_i^j = x_i^{j+1} - x_i^j$ (and similarly for $\tilde{m}_i^j$). The relative error is calculated as $e_i^j = |m_i^j - \tilde{m}_i^j|/m_i^j$, and the absolute error as $E_i^j = |m_i^j - \tilde{m}_i^j|$. These error values represent how different the frame-to-frame motion of the object is between the two simulations.

In interpreting these error measures, note that the individual e_i^j and

timing difference is not uniform over the course of the simulation. As one would expect, the performance improvement occurs when the objects start to pile up. When piles consistently grow (as in Figure 5), the performance improvement continues to grow. When a pile reaches a maximum size (as in Figure 6), the performance improvement tends to level off, though it remains significant.

Comparing timings of the Bunny Matrix examples, we notice that the more piling (i.e. the greater the angle of repose), the greater the speedup. This is to be expected, since objects that are in the pile are deactivated more readily.

Our timing comparison is somewhat unfair, in that the full simulation does not deactivate any of the objects, even though this is a well-known tool within rigid body simulation. Typically, objects are put to "sleep" based on when their linear and angular velocity drops below some threshold. This is trickier in practice than it sounds, and there can be serious problems in piling, as objects are repeatedly made to sleep and wake up when new objects are piled on. To get a sense of how our more aggressive piling-based sleep method compares to a deactivation method based strictly on velocity, we analyze the performance of the Bullet engine [Coumans and et. al], which has an efficient implementation of object sleeping. Specifically, we compare Bullet with and without object sleeping, on the Bunny Matrix example (using the default physics). The difference obtained by allowing sleeping objects in Bullet is a 1.09x

E_i^j are unlikely to be helpful to consider in isolation, since a single particular object might behave quite differently between two simulations, although the simulation as a whole might be very similar. What is of more interest is the degree to which the change in simulation method creates a global change in the character of the simulation. This can be inferred by averaging the e_i^j or E_i^j over all i, which will give a measure of the overall amount of difference in motion between the two simulations; note that the absolute error roughly corresponds to average momentum, and thus does not have a meaningful scale in general. We report both relative and absolute error for two examples, shown per frame in Figures 5 and 6, and relative error averaged over all frames in Table 1.

In examining the error rates over the course of simulations, we see that the error rates fluctuate significantly. Notice that while the error in Examples 5 and 6 are approximately the same on average, they behave quite differently during the simulation. Note also (e.g. end of Figure 5) that the relative error becomes a less reliable measure as the simulation comes to a near-rest (the m_i^j denominators in e_i^j approaching 0). We notice that the peak errors in Figure 6 tend to occur at points of significant overall motion (i.e. when rings "splash" out of the box).

5 Conclusion

Our approach for simulating piles of objects offers a simple method for representing one of the more interesting features of large rigid body simulations. By representing the piles explicitly, we can deactivate simulation of objects within the pile, enabling noticeable speedups. Designers of a rigid body simulation can achieve a desired object piling effect without implementing a more complex and slower frictional model component. In this way, we not only allow for a richer variety of simulations, but we achieve this with somewhat faster performance.

There are some clear limitations to our approach. If we simulate with multiple objects whose volume varies significantly (e.g. marbles and basketballs), the way a pile should form is difficult to specify, and our method would not perform well. Small variations (e.g. a factor of up to 2 or somewhat more) in volume should not be an issue, however (see video example). Also, our approach works more for random piles, rather than structured stacking (see Erleben [2007] for an approach for structured stacking).

There are also interesting avenues open for future work. We have proposed here an error metric for capturing the global change in behavior for different simulations. While we believe this is an interesting first step in that direction, it seems likely that better metrics for measuring error should exist. Further, although our method has been developed and tested in a rigid body simulation context, there is no fundamental reason why it could not be extended to handle deformable objects, as well. Examining the performance in such situations, or in other contexts such as statistical simulation [Hsu and Keyser 2009], would be an interesting direction for future work.

Acknowledgements

References

BARAFF, D. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 223–232.

BARAFF, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 23–34.

COUMANS, E., AND ET. AL. Bullet physics library. Open source: bulletphysics.org.

DRUMWRIGHT, E. 2008. A fast and stable penalty method for rigid body simulation. *IEEE Transactions on Visualization and Computer Graphics 14*, 1, 231–240.

EHMANN, S. A., AND LIN, M. C. 2001. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In *in Computer Graphics Forum*, 500–510.

ERICSON, C. 2005. *Real-Time Collision Detection*. Morgan Kaufmann.

ERLEBEN, K. 2007. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph. 26*, 2, 12.

FEARING, P. 2000. Computer modelling of fallen snow. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 37–46.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Nonconvex rigid bodies with stacking. *Transactions on Graphics 22*, 871–878. Proceedings of SIGGRAPH 2003.

HAHN, J. K. 1988. Realistic animation of rigid bodies. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 299–308.

HSU, S. W., AND KEYSER, J. 2009. Statistical simulation of rigid bodies. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, 139–148.

KAUFMAN, D. M., EDMUNDS, T., AND PAI, D. K. 2005. Fast frictional dynamics for rigid bodies. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM, New York, NY, USA, 946–956.

KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. 2008. Staggered projections for frictional contact in multibody systems. In *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA)*.

LIU, L., AND ZHOU, L. 2008. Numerical study on sandpile formation of granular materials with different grain size distributions. *Geotechnical Engineering for Disaster Mitigation and Rehabilitation*, 374–380.

MIRTICH, B., AND CANNY, J. 1995. Impulse-based dynamic simulation. In *WAFR: Proceedings of the workshop on Algorithmic foundations of robotics*, A. K. Peters, Ltd., Natick, MA, USA, 407–418.

OTADUY, M. A., TAMSTORF, R., STEINEMANN, D., AND GROSS, M., 2009. Implicit contact handling for deformable objects, apr.

SCHMIDL, H., AND MILENKOVIC, V. J. 2004. A fast impulsive contact suite for rigid body simulation. *IEEE Transactions on Visualization and Computer Graphics*, 2004.

ZHOU, Y. C., XU, B. H., YU, A. B., AND ZULLI, P. 2002. An experimental and numerical study of the angle of repose of coarse spheres. *Powder technology 125*, 45–54.