Automatic Simplification of Particle System Dynamics

David O'Brien Susan Fisher Ming C. Lin
Department of Computer Science
University of North Carolina at Chapel Hill
{obrien,sfisher,lin}@cs.unc.edu
http://www.cs.unc.edu/~geom/SLOD

Abstract

We present a novel framework for automatically simplifying the dynamics computation of particle systems to improve simulation speeds. Our approach is based on a physically-based subdivision *scheme to generate a hierarchy of* approximated motion models *or* simulation levels of detail (SLOD). *At each time step, the SLODs are updated on the fly and the appropriate SLOD is chosen adaptively to reduce computational costs. We have tested a prototype implementation on the simulation of a water fountain and a galaxy system. The preliminary results show a significant performance gain on these scenarios with little loss in the visual appearance of the simulation, indicating the potential to generalize this approach to other dynamical systems.*

1. Introduction

A key component for real-time interaction with virtual environments or computer games is the fast computation of appropriate, realistic behavior for moving objects based on the laws of physics. Despite recent technological advancements in graphics hardware and processor technology, current algorithms and systems still can not simultaneously simulate *real-time* dynamics for *large, complex* environments. Several rendering acceleration techniques have been proposed to improve rendering frame rates. These include visibility culling, model simplification and image-based rendering, etc.

One possible way to address the problem of generating and displaying complex dynamical systems is to design "simulation acceleration techniques" similar to those used for rendering acceleration. Recently, various hand-designed heuristics have been proposed to reduce computational costs for some specialized simulation scenes. In this paper, we consider the design of a simulation acceleration technique for the automatic generation and selection of *simplified and approximated* motion models based on application requirements. These application requirements may include user interests, critical events in a simulation, storage requirements, or variable accuracy requirements for different parts of a simulation.

The simulation acceleration technique we address in this paper is closely related to the concept of model simplification for rendering acceleration. Model simplification uses geometric levels of detail or a multi-resolution hierarchy for computing and utilizing a correspondence between the original model and a simplified one. A simplified model is used for rendering when the object occupies a small portion of the displayed scene, while the original model is used only when a close-up view of the object is required. This technique can substantially reduce the number of polygons rendered and the geometric processing required and thereby improve rendering frame rates. A similar approach for simulations, in which a simplified motion model is used when the frame rate drops below an acceptable threshold, may likewise provide performance gains for interactive simulations. We call this approach *dynamics simplification*.

Although there are existing simulation acceleration techniques such as [4, 5, 15, 24], there is no known algorithm for the *automatic* dynamics simplification of complex physical or biological systems. In this paper, we limit the scope of our investigation to particle systems as a first step toward the design of automatic dynamics simplification. Particle systems are commonly used in computer graphics, physically-based modeling, and animation of natural phenomena and group behavior. Additionally, they are often the building blocks of highly complex dynamical systems. Therefore, we hope the results of this paper can lead to the generalization of dynamics simplification for a broader class of dynamical systems.

1.1 Main Contribution

In order to achieve automatic dynamics simplification of a particle system, a mechanism for generating a hierarchy of approximated motion models or simulation levels-of-detail (SLOD) must be designed. We propose to automatically construct the SLODs for a particle system, by using a *physically-based subdivision* scheme to cluster particles. We then approximate each cluster of particles as one weighted particle in the force computations, and apply the result to all particles in the cluster.

At each simulation step, the hybrid tree is updated based on changing simulation requirements as defined by the user or by the nature of simulation. Appropriate SLODs are adaptively generated based on this subdivision given the requirements for different regions of the simulation and a desired minimum frame rate.

We have implemented a prototype system that can automatically generate simplified motion models for a particle system, select appropriate SLODs, and switch between them seamlessly. We have applied our framework to the dynamic simulation of a water fountain and an N-body system. The preliminary results show a substantial performance improvement on these test scenarios, with little loss in the visual appearance of the simulation.

1.2 Organization

The rest of the paper is organized as follows. Section 2 gives a brief survey of related work. Section 3 presents an overview of our approach. Hierarchy construction for simulation levels-of-detail is described in section 4. Next, the mechanism for automatic switching and selection of SLODs is discussed in section 5. Section 6 presents the results of our prototype implementation, and compares its performance against the traditional simulation methods. Finally, we conclude with future research directions in section 7.

2 Related Work

In this section, we first briefly describe prior work on geometric algorithms for LOD generation and multi-resolution modeling techniques, then survey other related research on dynamics simplification.

2.1 Model Simplification

Model simplification algorithms, such as geometric level-of-detail and multi-resolution modeling techniques, have been proposed to accelerate rendering of complex geometric models [8, 7, 18, 11, 25]. A recent survey on polygonal model simplification is available [20]. Funkhouser and Sequin described a generic framework for selecting and switching between different geometric levels-of-detail (LODs) to attain a nearly constant frame rate for real-time walkthrough of architectural buildings [10].

2.2 Dynamics Simplification

Compared to the rich body of literature on rendering acceleration techniques, there is relatively little work proposed on the use of LODs or multi-resolution modeling techniques for physically-based object manipulation, interaction and simulation.

Pai and Reissell developed a wavelet decomposition with a 2D bounding volume hierarchy for manipulating image curves in the plane [22]. Hirota, et al. presented an efficient algorithm for volume-preserving free-form deformation using a multi-level optimization algorithm for physically-based model manipulation and deformation [16, 17]. Recently, Ehmann and Lin proposed the use of geometric level-of-detail for accelerating general proximity queries between convex polyhedral models [9].

Motion models or simulation levels of detail can be generated from either pre-recorded motion sequences, procedurally approaches, kinematics, or based on dynamics computation. Some of the earlier human motion models in computer animation exploited this concept implicitly by using procedurally generated motion, simplified dynamics and control algorithms, off-line motion mapping, or motion play-back [2, 3, 6, 12, 13, 19, 23].

Carlson and Hodgins explored techniques for reducing the computational cost of simulating groups of legged creatures when they are less important to the viewer or to the action in the virtual world [4]. In this work, the generation of simulation LODs, switching and selection are designed by hand for a group of legged creatures. However, their experimental results are indicative of the potential of automatic simplification of general dynamical systems.

By using the spacetime constraint dynamics formulation, Popovic and Witkin introduced a motion transformation technique that preserves the essential properties of animated character motion with drastically fewer number of degrees of freedom [24]. Multon, et al. suggested a series of simplified walking models for mobilizing on complex terrain, as well as how and when the transition takes place [21].

Other types of simulation acceleration techniques have also been investigated to reduce the total computational simulation costs for a large, complex dynamical system. Chenney, et al proposed view-dependent culling of dynamic systems to speed up the computation of dynamics by ignoring what is not visible to the viewer [5], similar to view culling. Neuro-Animator, trained off-line using neural networks and motion trajectories generated by physic-based models, can yield realistic animations [15], resembling approaches from image-based rendering and modeling.

Finally, the scientific community also uses simulation acceleration techniques to make large computational problems, such as N-body simulation, tractable [1, 14]. The goal

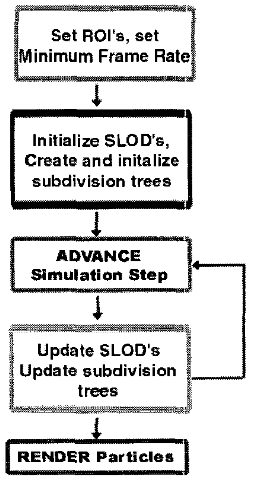

Set ROI's, set Minimum Frame Rate
Initialize SLOD's, Create and initalize subdivision trees
ADVANCE Simulation Step
Update SLOD's Update subdivision trees
RENDER Particles

Figure 1. System Architecture of Our Dynamics Simplification Framework.

of these highly evolved methods is very accurate simulation with quantifiable errors at non interactive rates. This differs significantly from our goal of realistic physically-based animations at interactive rates.

3 Overview

In this section, we give an overview of our approach. Given a target frame rate and target fidelity requirements defined by the user and the simulation, our technique is built upon a *physically-based subdivision* of the entire particle system. The resulting subdivision hierarchy is used to generate *simulation levels of detail (SLODs).* As the simulation proceeds, the subdivision hierarchy **(SLOD-tree)**, and thus the SLODs, are updated on the fly . The resulting computation is then displayed graphically to the viewer. Fig. 1 shows the overall system architecture of our dynamics simplification framework.

We will initially state our assumptions regarding the types of particle systems we consider and the challenges these systems present to our acceleration technique. We then briefly describe our techniques for dynamically updating particle clustering as the simulation advances, and how physical properties can be handled as extra dimensions in our SLOD-tree to generate more accurate SLODs. Then we give an overview on switching between different SLODs.

3.1 Assumptions and Approaches

Particle systems generally exhibit a high level of spatial coherence, meaning nearby particles behave in approximately the same way. Our basic approach is to group the particles into clusters based on spatial proximity and physical parameters, and then to compute the dynamic simulation on only the particle clusters. The behavior of the cluster center is uniformly applied to each particle in the cluster. In an ideal situation, if we could group ten particles into each cluster, our simulation should run almost ten times as quickly in an application that uses linear-time force computation models.

We consider two general kinds of particle systems, field systems and N-body systems. A field system is one in which all forces can be applied to individual particles in constant time. The total running time is linear with respect to the number of particles. In an N-body simulation, particles apply force to each other, resulting in $O(N^2)$ force calculations. Our N-body simulation is based on the well known Barnes-Hut Algorithm [1], which approximates these force calculations in $O(NlgN)$ time. For each type of particle system, we use an SLOD-tree to partition the particles into clusters. In general, this is an $O(NlgN)$ operation, whose cost must be amortized over several simulation steps.

3.2 Dynamic Spatial Subdivision

If a complete SLOD-tree were calculated at each simulation step, the overhead could easily dominate a simple field particle system. To overcome this, we do a complete subdivide once and then at each simulation step make only incremental updates to the SLOD-tree and to the location of particle clusters in the SLOD-tree. This occurs when a particle cluster moves across a subdivision boundary into an adjacent cell in the SLOD-tree. We use axis-aligned cells in our subdivision scheme, so computing when a particle cluster crosses a cell boundary is a fast calculation that we can perform often. This scheme allows for the addition, deletion, and maintenance of particle clusters as they move between *areas* requiring different resolutions of accuracy. These areas are called the *regions of interest* (ROIs) and are described in more detail in section 4.

3.3 Subdivision Based on Physical Parameters

We generalize the subdivision of particles using physical parameters (such as mass, velocity, acceleration, etc.) in addition to the standard Cartesian position. For example, faster moving particles could be grouped into smaller clusters for more accurate calculations. This is more efficient than forcing all particles into small clusters. In general, the above dynamic subdivision algorithm can be extended by treating physical parameters, as though they were additional spatial dimensions. However, in practice, the exponential increase in the number of cells for each added dimension limits this direct approach. We consider additional physical parameters as extra dimensions for partitioning at only the top levels of subdivision, and then proceed to subdivide on the three spatial dimensions.

3.4 Defining and Switching between SLODs

Each ROI defines several partitioning parameters that determine the how large the particle clusters are in that region. The various sizes of the resulting clusters then define the SLODs. The particle dynamics of each particle in a cluster is replaced by the particle dynamics of its weighted center of mass. When ROI partitioning parameters cause the SLOD-tree to produce a smaller number of larger clusters, the result is a coarser SLOD. Likewise, when the SLOD-tree generates many clusters, we have a finer SLOD in that region. ROI partitioning parameters can be updated automatically as the dynamical system evolves or by user input. This provides the mechanism for dynamically switching the SLODs and is how we maintain a target frame rate. Details for switching between SLODs are given in section 5.

4 Automatic Generation of SLODs

The physically-based subdivision hierarchy, or SLOD-tree, is the basis for generating SLODs for particle system dynamics. We use this novel hybrid hierarchy that encodes the definition of a simulation level of detail into the subdivision rules. We create separate areas in the simulation that progress at different resolutions of accuracy. These are called **regions of interest (ROIs)** and are defined either by the *user* (e.g. center of viewing frustum) or by *critical events* (such as collisions) that may occur in the simulation. These ROIs may move, resize and change parameters as the simulation proceeds. An SLOD-tree is used to subdivide the particles appropriately in each ROI into clusters to simplify the dynamics of the particle system. These particle clusters thus define the corresponding SLOD associated with each ROI.

For example, the center of view frustum may be given highest priority. The dynamics of the particle system in this ROI is then computed with little or no simplification to maintain high accuracy visual fidelity. Peripheral areas of the view frustum may be given secondary importance. Larger particle clusters are used in the simulation calculations to create an intermediate simulation level of detail. Finally, for the ROIs completely outside of the view frustum, the simulation may be computed with the lowest level of detail by clustering together tens, or even hundreds, of particles. This hierarchical subdivision scheme allows several concurrent and dynamically defined SLODs.

Next, we describe some basic partitioning parameters used in the SLOD-trees and detail how an SLOD-tree is used to perform adaptive subdivision of particles on the fly, then describe how SLODs are generated given the particle clusters.

4.1 Defining Partitioning Parameters

A geometric level of detail is often defined by the number of polygons in a scene. With SLODs for particle system dynamics, the corresponding measure is the number of particle clusters simulated in a scene, or inversely, the number of particles per cluster. We therefore define one primary partitioning parameter, *maximum cluster size*, as the maximum number of particles allowed per cluster. Any cell in the SLOD-tree whose cluster contains more than this number of particles will be subdivided into two cells each with a cluster of about half the size. The maximum *spatial* size of particle clusters, *maximum cluster breadth*, must also be limited to preserve visual and simulation integrity. The SLOD-tree is built so that the dimensions of each cell is no larger than this value.

These two parameters are not enforced as hard constraints, but rather serve as goals for the subdivision mechanism to maintain. For example, if two clusters can be combined into a new one with a particle count very close to the maximum size, they will be combined even if the breath of the new cluster may temporarily exceed the maximum cluster breadth. Furthermore, it is nearly impossible to generate a set of clusters that all have very close to the maximum number of particles, because we normally must divide clusters in half when building or updating the SLOD-tree. We try to build clusters of maximum size when possible by merging smaller clusters, but we realize that in practice the average cluster size will be somewhat less than the maximum.

Additionally, our hierarchy construction algorithm allows the definition of parameters that relate to physical properties. Although these constraints are not always used in constructing the subdivision hierarchy, they are crucial to deciding when particles or clusters can be grouped when performing updates to the subdivision. For example, if we chose not to use speed as a dimension for tree construction, we can instead include the parameter that limits clusters from combining unless their relative speeds are within a certain ratio. The velocity vectors of two clusters can also be constrained to be within a certain angle before the clusters are allowed to merge. This method is very fast and very memory efficient.

4.2 SLOD-Trees

Our SLOD-tree is a hybrid subdivision tree that combines k-D trees and uniform grids. K-D trees are attractive for particle systems because they are able to subdivide adaptively and produce finer division in areas of greater particle density. k-D trees handles the division of close particles extremely well, and in general produces far fewer empty leaf nodes than do oct-trees. Uniform grids are attractive because placement of a particle into a grid is very quick compared to the time needed to recurse down a k-D tree.

The top level of our SLOD-tree consists of a master node (as shown in Fig. 2 and Fig. 3), which holds pointers to multiple subtrees. One subtree is created for each ROI. Subtrees are based on a uniform grid structure. When a finer resolution of accuracy is needed, each cell in a subtree uniform grid can serve as the root of a k-D tree. As the simulation progresses, these k-D trees are expanded and pruned to contain only as many cells as needed. Each subtree divides independently according to the partitioning parameters, as defined above, of its associated ROI. The uniform grid is initialized so that each grid cell is no larger than the maximum cluster breadth. New subtrees for dynamically created ROIs may be added to the master node at any time during the simulation.

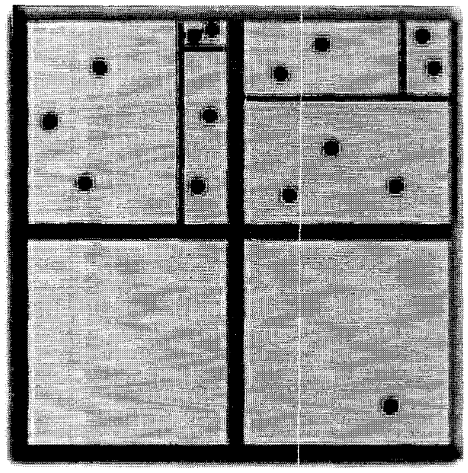


Figure 2. A subtree is made from a uniform grid of cells with the grid size set to the maximum cluster size. When a uniform grid needs to be further subdivided, it follows a k-D tree strategy.

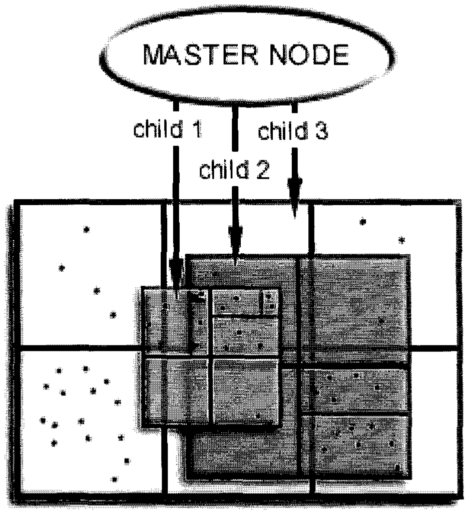


Figure 3. When a new particle is inserted into the simulation system, a master node sequentially queries each subtree from the top down until it finds the proper subtree for the new particle.

4.3 Subdivision Update

When a new particle is added to the system, or when a particle cluster travels across a cell boundary, the tree needs to be updated. If there are overlapping ROIs, the system must decide into which subtree to place the new particles or cluster. The master node maintains a precedence ordering of its children. Subtrees associated with higher resolution ROIs are normally placed in the front of the chain, followed by lower resolution subtrees, and ending with the subtree for the background, as shown in Fig. 3. When a master node receives a particle, it checks that particle against the boundary of each of subtree in sequential order. If the particle is found to be within the boundary of the first subtree it is sent there for insertion. If not, it continues to check subsequent children subtrees until the appropriate one is found.

Similarly when clusters cross grid cell boundaries, each cluster must query the SLOD-tree for its new location. Spatial coherence dictates that the proper cell will be found in the SLOD-tree very close to the previous one. In this case we use a bottom-up search rather than the top down approach for new cells. The cluster first checks whether it lies inside the boundary of its parent. If not, it continues to move up the tree until it finds a cell whose boundary it is within. Then the cluster begins a top down search from that enclosing node.

Unfortunately, a bottom-up search may fail to find the correct cell if there are overlapping subtrees. A cluster in the background ROI moving into a ROI of higher resolution (or greater importance) may insert itself into a nearby low resolution cell in the background subtree, rather inserting itself into the higher resolution on top of the background. than the higher resolution sub-tree. To compensate, we occasionally have to perform a top down search. Alternatively, we could use the faster bottom-up searching and simply try to avoid overlapping ROIs, but this approach would limit the use of ROIs that move or change size.

4.4 Simulation Levels of Detail

We simplify the dynamics computation for each particle cluster by approximating its particle system dynamics using the particle dynamics of its weighted center of mass. Given each particle cluster C consisting of n particles, we generate the corresponding SLOD or approximated motion model using the following procedure:

1. Compute the position P_{COM} and velocity V_{COM} of the weighted center of mass COM by

$$P_{COM} = \frac{\sum_{i=1}^{n} m_i P_i}{\sum_{i=1}^{n} m_i}$$

$$V_{COM} = \frac{\sum_{i=1}^{n} m_i V_i}{\sum_{i=1}^{n} m_i}$$

   where m_i, P_i and V_i are the mass, position and velocity of the ith particle.

2. Update the position and velocity of the CoM using the standard particle dynamics.

3. Apply the change in position and velocity, Δp and Δv to all particles.

5 Switching between SLODs

In our dynamics simplification framework, switching between different SLODs is governed by both the desired minimum frame rate and the regions of interests defined by the user and the simulation.

5.1 Using Regions of Interest

As with geometric LODs, areas closer to the viewer and more centered in the viewing frustum require a higher resolution SLOD. Additionly certain critical events in a simulation need higher resolution to maintain simulation correctness and integrity.

For example, particles bouncing off an uneven surface tend to scatter in different directions. This effect is lost when a large cluster of particles interacts with the same surface, since all the particles in the cluster will move off in the same direction. When a large cluster is near an uneven surface, the simulation must switch to a higher resolution SLOD. The cluster should automatically decompose into smaller clusters or into individual particles. After the collision as these individual particles move away from the surface, those with similar motion, physical properties, and location may again be grouped into a cluster, as shown in Fig. 6. We have tested this principle in a simulation with two collision areas, Fig. 7. The cone on the right is in a higher resolution ROI and produces more realistic collisions. When the smaller clusters fall away from this cone, they begin to merge back into larger clusters.

Areas surrounding particle emitters also need a higher-resolution SLOD. If new particles are grouped too close to a particle emitter, they do not have time to naturally spread out before their velocities are averaged together. This results in a much narrower and visually less pleasing particle stream from the emitter. An alternative approach for emitters would be to use the velocity of the first particle placed in the cluster as the center of mass velocity. Additional particles would not average in their velocity vectors, causing the distribution of cluster velocities to resemble the distribution of the particle velocities. This approach should be used only in small areas for it is a less physically accurate model. When the cluster has moved beyond emitter ROI, it will revert to using the weighted average when inserting particles or merging with other clusters.

One can also use ROIs to increase simulation fidelity to clusters based on their physical rather than spatial parameters. We could define two ROIs that each cover the entire space, using one to cluster the lighter particles and one to cluster the heavier particles. If the forces in the simulation interact more noticeably with the lighter particles, the ROIs would be set up to place the lighter particles into smaller clusters than the heavier particles. ROIs provide a flexible and powerful method for maintaining fidelity in crucial areas of a simulation so that the overall appearance of the particle system is maintained. However, some particle systems may simply have too many crucial areas to avoid some loss of fidelity. If there were collisions happening everywhere, it would soon become impractical to cover all collision surfaces with special ROIs. This is a natural trade-off between speed and accuracy that SLODs share with other simplification techniques, such as model simplification.

5.2 Minimum Frame Rates

The system tries to maintain a consistent frame rate by dynamically switching between the SLODs whenever it notices that the actual frame rate has deviated from the target frame rate. It is important that any switching is smooth and does not cause a noticeable change in the flow of the simulation. A naive approach would be to repartition all particles whenever the frame rate dictates a switch of SLODs. This is impractical because any global change could cause a notable pause in the simulation and a large temporary drop in the frame rate.

Switches between SLODs are implemented indirectly by adjusting the clustering parameters in the ROI subtrees. When the frame rate is higher than necessary, the maximum cluster size for each ROI is decreased. This will cause the subtrees to begin breaking large clusters into smaller clusters whenever their positions in the SLOD-tree are updated. Reverse adjustments are made when the frame rate becomes too low. This approach does not create an instantaneous change in the number of clusters but a gradual fluctuation over several frames. If the system cannot sustain the desired frame rate after several adjustments to the maximum cluster size, the system will also adjust the maximum particle breadth and the grid size at the top level of each ROI subtree. This approach is more likely to cause a dip in the frame rate as every cluster in that subtree will have to be reinserted during the next update cycle.

6 Implementation and Results

We have implemented a prototype system in C++ to automatically generate SLODs for a particle system, select appropriate SLODs and switch between them. The simulation results are displayed using OpenGL. We have tested our system framework on two scenarios, a field system and an N-body system. With found that We observed up to two orders of magnitude performance improvement, with little or no loss in visual appearance of the simulation. No hand tweaking of parameters nor specialized low-level optimization was used to obtain the timing comparisons.

6.1 System Demonstrations

The first test scenario is a fountain with three streams of water particles. The simulation includes force calculations for gravity, air resistance, and a light wind. Each of these forces can be calculated in constant time so the simulation without SLOD should run in linear time with respect to the number of water particles. The three particle emitters were placed in very small ROIs that allowed very limited clustering. The simulation starts with only one stream and adds the remaining two streams at five second intervals. Each stream produces approximately 2000 particles. We ran the tests with and without SLODs and set the target frame rate to 30 fps. A snapshot of this simulation is shown in Fig. 9.

The simulation and rendering overhead of the fountain quickly limited the frame rate of the simulation without SLODs. It could not meet the target frame rate even with only one water stream, and continued to slow down noticeably with each increase in the number of particles. The simulation with SLODs was able to easily sustain the 30 fps target for the entire run. At the end of the simulation, it was running approximately six times faster than the one without SLODs. It maintained the desired frame rate by incrementing the cluster size for the ROIs near the particle emitters. The cluster size was never incremented above 30. To determine the appropriate adjustment to the cluster size, we used a simple scheme that incremented (decremented) the maximum cluster size by one, whenever the average of the previous three frames was below (above) our target framerate. For a simulation with more ROIs, we increment the maximum cluster size of each ROI by the same percent. This produces an even increase throughout the simulation. Fig. 8 plots the frame rate of both simulation runs.

The second simulation is a classic N-body simulation showing a galaxy of 10,000 stars rotating in space. A snapshot of the simulation is shown in Fig. 4. Our simulation only calculates exact forces between nearby clusters and uses approximations for forces from distant clusters. The accuracy of the algorithm can be tuned by varying the parameter that determines what is a distant node. Different from the Barnes-Hut algorithm typically used in a N-body simulation, our algorithm also considers particle mass in the partitioning step. Typically, light particles have little or no effect on more massive particles even when they are close by. By separating out the lighter particles into a separate ROI, we can reduce the force calculations by simply ignoring the light particles or by grouping them into much larger clusters. Clusters can simply bypass the lighter subtree when they calculate what forces affect them, while the lighter clusters traverse both subtrees.

In theory, the overhead of creating and updating SLODs can outweight the benefits, if the average cluster size is small. In order to measure when the benefits begin, the two simulations were set with static ROIs that made no adjustments to maintain frame rates. We set the maximum cluster size of the main ROI to different values and computed the average frame rate for the entire simulation. These rates were compared with the non-SLOD frame rate and converted to relative speed increases, which are summarized in Fig. 5.

For the fountain, SLODs started to yield faster frame rates when the maximum cluster size was approximately 6. The increases are roughly linear with respect to the number of particles per cluster, as we would expect given the simple force calculations of the fountain. The N-body simulation frame rates increased even more quickly and approached two orders of magnitude when the maximum cluster size

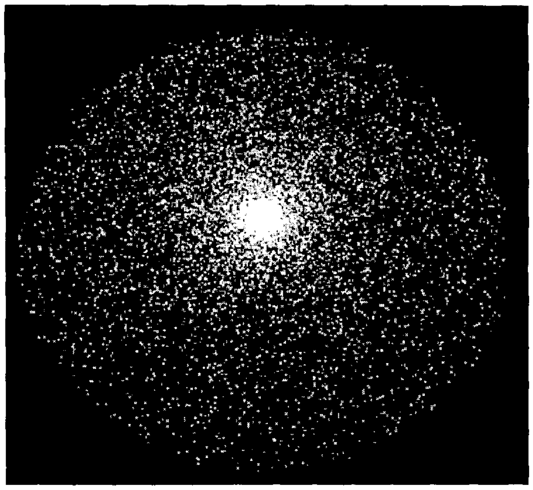
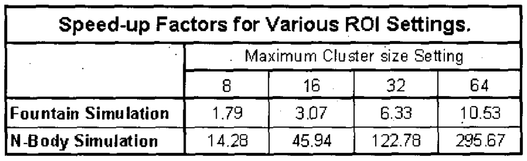


Figure 4. A Snapshot of the Galaxy System Simulation. SLODs greatly reduce the number of particle to particle force computations resulting in substantial performance gains.



Speed-up Factors for Various ROI Settings.				
	Maximum Cluster size Setting			
	8	16	32	64
Fountain Simulation	1.79	3.07	6.33	10.53
N-Body Simulation	14.28	45.94	122.78	295.67

Figure 5. Speed-up factor of simulations with various ROI settings compared to the simulations run without SLODs.

was set to 64. These results clearly demonstrate the potential of SLODs to accelerate complicated dynamical systems.

6.2 Error analysis

Error metrics for particle systems are difficult to define for all cases. A metric that works for one simulation may have little meaning in another. Our system was designed for computer animations and dynamics simulations where one is typically more interested in the *macroscopic* behavior of the particle system as a whole and the visual appearance of the particle simulation, rather than the accuracy of the simulation for each individual particle. Consequently, SLODs may have limited applicability in scientific computations in which accuracy of simulations is paramount.

We can bound the errors introduced by SLODs in certain cases. Recall that each ROI also has parameters limiting which clusters can be merged based on physical quantities. These are the maximum angle between their velocity vectors θ and the maximum speed ratio λ. Both the speed and direction of the two clusters will change when they are merged. By setting these values low, we can limit the change in speed and direction of a cluster when it is merged into another. Given two clusters α and β with masses α_{mass} and β_{mass}, normalized directions α_{dir} and β_{dir}, and speeds α_{sp} and β_{sp}, we can calculate the change in speed Δ_{sp} and direction Δ_{dir} for cluster α as:

$$\Delta_{sp} = \frac{\beta_{mass}}{\beta_{mass} + \alpha_{mass}} \alpha_{sp}(\lambda - 1)$$

$$\Delta_{dir} = \frac{\beta_{mass}}{\beta_{mass} + \alpha_{mass}} \theta$$

These formulas show that the deviation on a cluster increases as it merges with clusters of greater mass. To reduce errors in a simulation that uses particles with a large variation in mass, the simulation should benefit if one subdivides with mass as a dimension, as described in section 3.3. This will prevent lighter clusters from merging with heavier ones and reduce the frequency of merges that produce larger deviations.

It is difficult to quantify the cumulative effect of these merges. However, the spatial coherence of most particles systems leads us to believe that they will not introduce noticeable artifacts in the rendered appearance of the simulation results. This can be confirmed visually by viewing side by side videos of our simulations of the fountain and N-body system, which are available at our project website:

http://www.cs.unc.edu/~geom/SLOD/

7 Summary and Future Work

In this paper we present a new method for automatic simplification of particle system dynamics using a physically-based subdivision scheme, which considers the physical properties of the dynamical system, user's intents and the critical events of the simulation. In comparison to traditional simulation techniques, we observed performance gains approaching two orders of magnitude. This system is able to maintain the desired simulation frame rate. with little degradation in the resulting visual display of the simulation.

The promising results indicate the potential of dynamics simplification. There are several open research issues. Can we generalize this approach to other dynamical systems? If so, is there a systematic mechanism to derive maximum performance gain by using SLODs? If not, what other techniques should we consider? How do we ensure the integrity of the simulation is not lost and the simulation results are correct (or at least consistent) using dynamics simplification? The design of other simulation acceleration techniques also presents similar new challenges.

References

[1] J. Barnes and P. Hut. A hierarchical $O(n \log n)$ force-calculation algorithm. *Nature*, 324(4):446–449, Dec. 1986.

[2] A. Brudlerlin and T. Calvert. Goal-directed, dynamic animation of human walking. In *Computer Graphics (Proc. of SIGGRAPH'89)*, pages 233–242, 1989.

[3] A. Brudlerlin and T. Calvert. Knowledge-driven, interactive animation of human running. In *Proc. of Graphics Interface*, pages 213–221, 1996.

[4] D. Carlson and J. Hodgins. Simulation levels of detail for real-time animation. In *Proc. of Graphics Interface 1997*, 1997.

[5] S. Chenney and D. Forsyth. View-dependent culling of dynamic systems in virtual environments. In *Proc. of ACM Symposium on Interactive 3D Graphics*, 1997.

[6] C. A. Chrislip and J. F. Ehlert. *Level of detail models for dismounted infantry in NPSNET-IV.8.1.* M.S. Thesis, Naval Postgraduate School, 1995.

[7] J. Cohen, M. Olano, and D. Manocha. Appearance preserving simplification. In *Proc. of ACM SIGGRAPH*, pages 115–122, 1998.

[8] J. Cohen, A. Varshney, D. Manocha, and G. T. et al. Simplification envelopes. In *Proc. of ACM Siggraph'96*, pages 119–128, 1996.

[9] S. Ehmann and M. C. Lin. Accelerated proximity queries between convex polyhedra using multi-level voronoi marching. *Proc. of IROS*, 2000.

[10] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 247–254, Aug. 1993.

[11] M. Garland and P. Heckbert. Surface simplification using quadric error bounds. *Proc. of ACM SIGGRAPH*, pages 209–216, 1997.

[12] M. Girard and A. Maciejewski. Computational modeling for computer animation of legged figures. In *Computer Graphics (Proc. of SIGGRAPH)*, volume 19, pages 263–270, 1985.

[13] J. Granieri, J. Crabtree, and N. Badler. Production and playback of human figure motion for 3d virtual environments. In *Proc. of VRAIS*, pages 127–135, 1995.

[14] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, Aug. 1987.

[15] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. *Proc. of SIGGRAPH*, pages 9–20, 1998.

[16] G. Hirota, R. Maheshwari, and M. Lin. Fast volume-preserving free-form deformation using multi-level optimization. *Proc. Symposium on Solid Modeling and Applications*, 1999.

[17] G. Hirota, R. Maheshwari, and M. Lin. Fast volume-preserving free-form deformation using multi-level optimization. *Computer-Aided Design*, 32(8 & 9):499–512, 2000.

[18] H. Hoppe. Progressive meshes. In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, Aug. 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[19] H. Ko and N. Badler. Straight-line walking animation based on kinematic generalization that preserves the original characteristics. In *Proc. of Graphics Interfaces*, 1993.

[20] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygon environments. In *Proc. of ACM SIGGRAPH*, 1997.

[21] F. Multon, B. Valton, B. Jouin, and R. Cozot. Motion levels of detail for real-time virtual worlds. *Proc. of ASTC-VR'99*, 1999.

[22] D. K. Pai and L. M. Reissel. Haptic interaction with multiresolution image curves. *Computer and Graphics*, 21:405–411, 1997.

[23] K. Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):5–15, 1995.

[24] Z. Popovic and A. Witkin. Physically based motion transformation. In *Proc. of SIGGRAPH 1999*, pages 11–20, 1999.

[25] P. Schröder and D. Zorin. Subdivision for modeling and animation. *ACM SIGGRAPH Course Notes*, 1998.

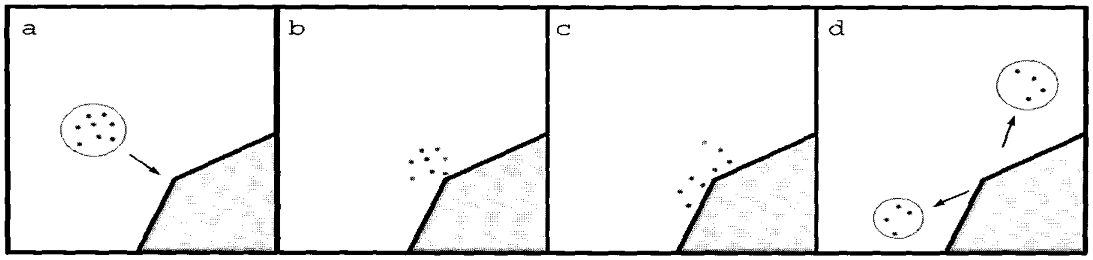


Figure 6. (a) A large cluster approaches an uneven surface. (b) As it nears the surface, it is switched to a higher resolution SLOD. (c) This is used until the particles move away from the surface when (d) the simulation reverts to a lower resolution SLOD.

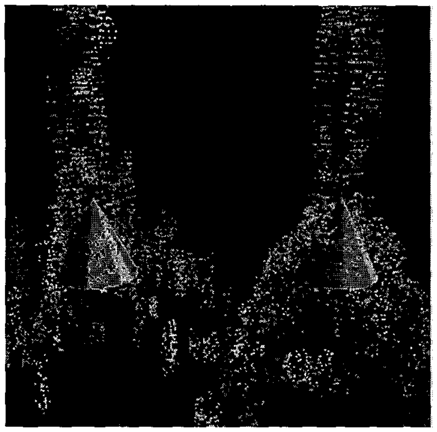


Figure 7. Particles collide with two cones. The right cone is surrounded by a higher resolution ROI and produces better looking collisions.

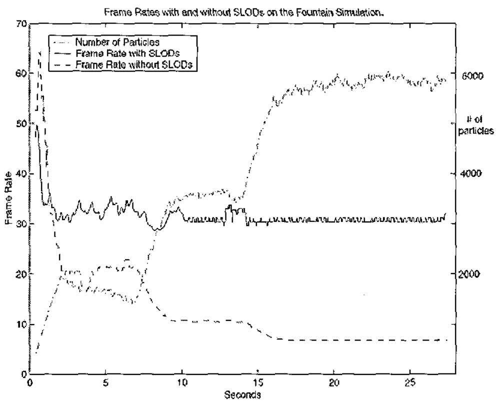


Figure 8. Frame rates of the fountain simulation with SLODs (solid black line) and without (dashed blue line). The SLOD version stays close to the target frame rate of 30 fps. Without SLODs, the rate rises in proportion to the number of particles (red).

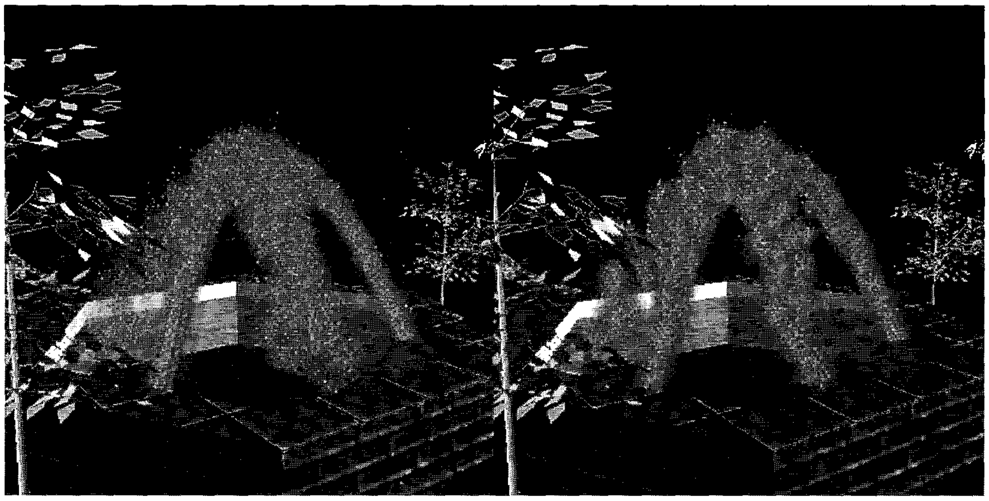


Figure 9. Two snapshots of the fountain simulation. The left simulation uses no SLODs, while the right one uses SLODs with a maximum cluster size no greater than 30. The SLOD simulation runs 6 times as fast and maintains a consistent frame rate.