Appears in the Proceedings of the 1997 Symposium on Interactive 3d Graphics, Providence, RI, April 27-30 1997

View-Dependent Culling of Dynamic Systems in Virtual Environments

Stephen Chenney David Forsyth*

University of California at Berkeley

Abstract

Scalable rendering of virtual environments requires culling objects that have no effect on the view. This paper explores culling moving objects by not solving the equations of motion of objects that don't affect the view. While this approach could be scalable for many kinds of environments, it raises two problems: consistency - ensuring that objects that come back into view do so in the right state - and completeness - ensuring that objects that would have entered the view volume as a result of their motions, do so.

Solutions to these problems lie in studying the statistics of the motion of objects. We show strategies for addressing the problem of consistency, with a number of examples that illustrate both the difficulties involved in, and the potential gains to be obtained by, formulating a comprehensive approach.

CR Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism - *Virtual reality*; I.6.5 [**Simulation and Modeling**]: Model Development - *Modeling methodologies* I.6.8 [**Simulation and Modeling**]: Types of Simulation - *Animation*

1 Introduction

Dynamic systems are an important component of virtual environments. However, traditional dynamic systems require constant computation to keep their state up to date. As virtual worlds become larger in size, we expect the number of systems with visible impact to grow at a slower pace, and less of the total dynamic state to be relevant for rendering a given view. To achieve maximum scalability, the cost of computing dynamics should depend only on what is relevant to the view, not on the total size of the world.

Level of detail [8] and visibility culling [2, 7] achieve scalability for static worlds by rendering only what is relevant. While Hodgins and Carlson[5] reduce the cost of dynamic simulation by using level of detail for hopping robots playing a chase-and-capture game, their approach requires distinct, hand generated models for the robot dynamics. Setas et. al. [9] present a range of models for trees moving in the wind, each corresponding to a different quality and cost of simulation, but there is no discussion of how dynamic state is maintained when trees are culled. In this paper we examine the implications of culling dynamic systems, and present examples to demonstrate possible modeling techniques.

*[schenney|daf]@cs.berkeley.edu

2 Culling Dynamics

Culling dynamics that have no visible impact offers to achieve scalability in virtual worlds, but results in two significant problems.

- **Consistency:** if the view turns away from a system of moving objects whose dynamics are then culled, in what state should those objects be when the view turns back? In particular, in an ideal system, an observer should not be able to obtain contradictions by looking away from an object, and then back at it.
- **Completeness:** objects that are out of the view volume often travel into it of their own accord - for example, consider a view into a room full of balls bouncing off walls. If the dynamics of a ball are culled as soon as it leaves the view volume, a fixed view could contain a steadily decreasing number of balls. While this could be perfectly consistent with evidence available to the viewer (there could be something out of view catching the balls), it is a poor model. Avoiding this difficulty requires some way of telling where an object might be, without necessarily solving its equations of motion.

A natural approach to the consistency problem involves tagging an object with the time it was last seen and its state at that time when it leaves the view volume, and then advancing its state (by solving the equations of motion with as large a time-step as possible) when the view volume moves back over the object. This is a poor solution for two reasons: it leads to sharp collapses in the frame rate as the view sweeps over a complex object which has been out of view for a while, and it offers no leverage on the completeness problem.

In our view, a better approach involves a study of the qualitative properties of dynamic systems. A viewer will detect an inconsistency if their qualitative prediction of system behavior is violated, so it is at this level that dynamic state must be consistent. Underlying our methods should be a model of human perception of dynamics that indicates which properties must be maintained, allowing us to avoid computing expensive, irrelevant properties.

A key property in designing a model for culling is the sensitivity of the system to initial conditions. A viewer has a snapshot of the approximate initial conditions when a system leaves the view, and it is the predictions they may make based on those conditions that determines how a new state is generated when the system re-enters the view. We characterize the influence of initial conditions in three ways, and associate different strategies with each.

- **Strong** influence allows accurate predictions to be made, leaving no choice but to integrate the system as if it were in view. However, this may introduce lag. As a partial solution to this problem we attempt to buffer state ahead of time, aiming to have state ready when the system re-enters the view. This does not save

work, but does spread the lag time over many preceding frames.

- **Medium** influence allows a viewer to make some qualitative predictions. There appears to be no completely general modeling strategy for this case. Instead we expect to find a set of techniques, each of which works for a class of dynamics, and apply those which best match the properties of a given system. One applicable technique is classical perturbation theory, as described in [1], which models how systems behave over time when subjected to small perturbations in control.
- **Weak** influence makes it difficult for a viewer to make any prediction about state based on their knowledge of initial conditions. However, predictions can still be made based on knowledge of the general behavior of the system - for example, energy or angular momentum might be conserved. To generate a new state in this scenario we encode the general behavior of the system using statistical or other means, then choose a new state consistent with our model.

We refer to this influence of initial conditions as *conditioning*. A key parameter in defining culling strategies is the time that must elapse before strong conditioning is replaced by medium and then weak. In the following sections we discuss some possible approaches to determining these time intervals such that the important qualitative measures of a system remain consistent.

We have implemented a virtual fun park environment in which to experiment with these ideas. The core system loops testing visibility and then updating the dynamics and rendering the geometry for visible objects only. The dynamic models are all designed such that they may be called at arbitrary time intervals as they move into and out of view. Here we examine two systems designed for use within this environment – the Tilt-A-Whirl and a bumper car ride.

3 The Tilt-A-Whirl

The Tilt-A-Whirl is an amusement park ride in which passengers sit in one of seven cars that are driven in unison around a hilly circular *track*. Each car is free to rotate about the center of a circular *platform* which tilts such that it remains tangential to the surface as it moves around the track. The key parameters of this system are the angular position of the platform on the track, θ, the angular position of the car on the platform, ϕ and the time derivatives of those variables, $\dot{\theta}$ and $\dot{\phi}$. The dynamics of the system moves through 5 phases:

- **Start**, where the platforms accelerate;
- **Run**, the steady state phase of the motion, with $\dot{\theta}$ fixed;
- **Stop**, during which the platforms slow down and stop;
- **Decay**, for which the platforms are stationary, but the cars are still in motion on their platforms;
- **Stationary**, where everything is static for some period.

The system loops through the phases in sequence driven by a state machine which changes state based on the time spent in each phase. We discuss the run and decay phases in detail, as they indicate underlying principles for designing systems with culling in mind. We then discuss how culling interacts with the state machine.

3.1 The Run State

In the run state, a Tilt-a-Whirl is unpredictable; culling systems like this requires a model of the general dynamic behavior from which we can generate new states. Frequently,

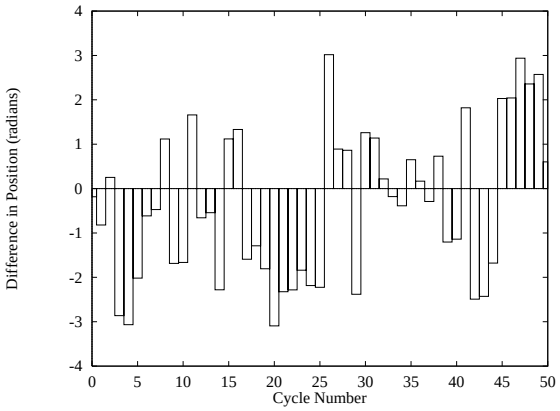

Figure 1: *A trace of the difference in ϕ for two cars started with similar initial conditions. The value of the difference is noise-like after a small number of cycles, meaning that the relationship between the 2 cars becomes difficult to predict after only a few cycles. In turn, this means the system's dynamics can be culled quite easily, because a viewer is unlikely to be able to make sufficiently accurate predictions to notice an error, if invariant properties of the motion can be preserved.*

systems of this form possess one or more attractors, to which the state of the system will move regardless of initial conditions. A an attractor in state space offers a useful model of the general behavior of the system – regardless of how the system has evolved, it will lie on an attractor.

The run phase of the Tilt-A-Whirl, when $\dot{\theta} = 6.5$rpm, exhibits chaotic steady state dynamics, which are described in detail in [6]. Figure 1 compares the motion of 2 cars that begin with similar initial conditions corresponding to a difference of $10°$ and $10°/s$ for ϕ and $\dot{\phi}$. The sequence indicates that after the cars have passed over 3 hills (which takes around 9s of virtual time), there is a wide variation in which car follows which, and by how much.

The Tilt-a-Whirl's attractor is an invariant property of the motion – any Tilt-a-Whirl that has been running for sufficient time has a state that lies on this set of states. Consider the probability that a Tilt-a-Whirl is close to state $\mathbf{x}$ at time t, given that it was observed to be close to some other state $\mathbf{x}_0$ at time t_0; because the Tilt-a-Whirl has an attractor, this probability is very low off the attractor, and because its behaviour is hard to predict, this value must be largely independent of $\mathbf{x}_0$ and t_0. As a result, we can model the Tilt-a-Whirl as a probability density on the attractor; when a car has been out of view and in the run state for some time, a sample from this density is a consistent model of the object's state.

3.2 The Decay State

In the decay phase, each car behaves like a damped pendulum. A good qualitative parameter to measure for the system is the total energy. The energy decays rapidly in a predictable way, and viewers can estimate it by noting the amplitude of the motion. In addition, below a key energy threshold the non-linear equation of motion may be linearized, to yield an explicit expression of state as a function of time.

Given an initial energy, we obtain an approximation to the time taken to reach the linearized region, at which point we can change from the non-linear to linear models. If the

system has been out of view for longer than this time, we can jump directly to the linear model and avoid the work of integrating through the non-linear phase.

3.3 Managing States

When a Tilt-A-Whirl system re-enters the view, we must first determine which phase the machine is in, and then determine a state for this system consistent with the last time the machine was seen. If the machine has not changed phase, we may use the methods above for the run or decay phase, or integrate forward for the other phases. If the system has changed phase, we step through the state machine to find the new phase, and then generate a new state across phases.

The stationary phase has the property that there is only one possible state for the system in this phase. Hence, we know the state of the system at least as recently as the last stationary phase. In addition, if the system has been through more than 9s of virtual time in the run phase, we can sample a state for the most recent moment in run with the knowledge that consistency is not violated. Finally, if the system has been in the decay phase for long enough to allow the equations to be linearized, we can generate a consistent state for that period. The longest time we will need to integrate over is the time to the most recent of the above events, which is always less than 20s virtual time or 80ms real time. The algorithm for generating a new state simply identifies the most recent event, and integrates from that point to the current time.

3.4 Results

We conducted a series of tests to compare the computational cost of dynamics with culling enabled to that with no culling. Three different types of viewer motion were examined corresponding to a fixed view, a slow, smoothly moving view and a view that moves rapidly and chaotically over the world. Figure 2 shows the significant speedups achieved with dynamics culling. The greatest speedups are for a static view, while the worst speedup is for fast moving views, which are unlikely to be used by a real viewer. The low slope of the culling curves indicates high scalability for the common case of walkthrough type views.

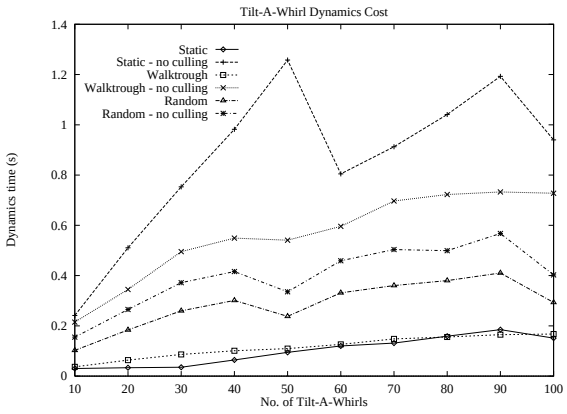

Figure 2: *Mean real time spent on dynamics per second of virtual time, plotted against the number of moving Tilt-A-Whirls for a variety of viewpoint movements, comparing culled dynamics with unculled dynamics. For a static view culling dynamics offers an order of magnitude speedup. For the slow moving walkthrough view the speedup is approximately a factor of five. The random view attaches the viewpoint to one of the Tilt-A-Whirl cars - which move quickly and nearly randomly - resulting in a speedup of about 1.5. This view performs poorly because systems are not out of view for long enough to allow the dependence on initial conditions to become weak. The Tilt-A-Whirls were placed such that the density of machines increased, not the size of the world. Note that the time spent on dynamics without culling varies because the integrator step length varies, which also contributes to the variations in slope.*

4 Bumper Cars

A bumper car simulation consists of 12 cars moving on an enclosed, flat platform. Each car is driven by a simple controller that attempts to drive the car around the track, avoiding the walls and other cars. Occasionally the controller becomes angry and deviates from its normal behavior to deliberately hit another car.

The controller for the car chooses a direction and speed in which the car would like to travel. It does this by choosing preferred parameters which take it in an elliptical circuit around the track at maximum speed. It then calculates a series of offsets to apply to these parameters in order to avoid the walls and other cars. The modified parameters are bounded to reflect the maximum acceleration of the car and the maximum turn rate of the steering wheel.

The continuous part of the dynamics simulation simply integrates the standard rigid body equations for the car. The steering wheel angle and drive motor speed exert forces on the system, and there are traction forces between the wheels of the car and the surface on which it travels. The controller is called at discrete intervals to provide a set of drive parameters, which are assumed to be constant through the next interval. The controller is deliberately bad at avoiding collisions. Collisions are tested at discreet intervals, using the OBBTree interference detection package as described in [3], and are treated as impulses [4].

The bumper car system is influenced by an initial state for a long period of time. For time intervals where the dependence on initial conditions is strong, we rely on buffering methods. The buffer is filled during periods without collisions, and empties as collisions occur.

The controller for each bumper car is such that it tries to produce a particular motion (movement at constant speed around an ellipse), but this constant motion is perturbed by outside factors such as avoiding neighbors. The exact spatial perturbations are not predictable by a viewer because aspects of the controller such as collision response, avoidance thresholds and random noise make perturbations sensitive to uncertainty in initial conditions. However, a viewer can detect the average effect of perturbations over time – the irregularities in a car's motion. This suggests that a good strategy is to capture the distribution of the spatial effect of the temporal mean of perturbations, sample a spatial effect and apply it to the constant elliptic motion expressed as an explicit function of time.

This model of perturbed motion on an ellipse suggests a parameterization of the state space: $(\phi, r, \rho, \dot{u}, \dot{v}, \dot{\rho})$, where ϕ defines the position of the car on an ellipse of major radius r and eccentricity matching the aspect ratio of the track. ρ encodes the orientation of the car with respect to the tangent to the ellipse at the car's location. The parameters $\dot{u}$ and $\dot{v}$ are the velocity of the car in its local coordinate system. The angular velocity of the car is encoded in $\dot{\rho}$. We build distributions on the effect of perturbations in ϕ, r and ρ independently.

Each car is sampled independently (the dependence be-

tween cars is accounted for in the perturbations), which may lead to cars in interference. A resampling strategy is not useful here, because the collision may be very likely to occur. Instead we reposition cars on the track such that the relative position of the car's centers is preserved if possible – only their separation distance is changed. This assumes that all collision avoidance or collisions proper result in a rebound type motion.

A model based on the effect of perturbations over time provides information for the long term behavior of the system. Intuitively, the initial uncertainty in the position of each car may be visualized as 12 small blobs in state space, each blob representing the (uncertain) state of one car. As time passes, the blobs move through state space as the cars move, but the uncertainty in the effect of perturbations causes them to grow in size, because we are unable to predict exactly how they move. Eventually the blobs spread out to cover all of state space, indicating that the cars may be anywhere on the track. In the limit case we expect the distribution of spread blobs after a long period of time to reflect the general behavior of the system in terms of the average position of cars on the track. The time at which we are close to this limit is the time at which the influence of initial conditions is sufficiently weak as to be ignored.

If sufficient time has passed for the initial state to be irrelevant, we independently sample the state of each car from a distributions on $(\phi, r, \rho, \dot{u}, \dot{v}, \dot{\rho})$. With this method it is possible to choose positions for cars that intersect either each other or the walls. We place cars in order and test for intersection with those already placed. If interference is detected, we reject and resample. This method produces distributions on cars that are qualitatively indistinguishable from dynamically evolved distributions.

4.1 Results

A series of tests was conducted to examine the performance improvements that are achievable when culling bumper cars. The experiments were identical to those for the Tilt-A-Whirl with bumper cars used in place of Tilt-A-Whirls. Figure 3 shows the speedups achieved for the bumper car world. The speedups achieved range from 5× to 1.5×, and the low slopes on the culling curve indicate a strong improvement in scalability.

5 Conclusion

The examples presented demonstrate techniques for ensuring that the qualitatively important properties of a dynamic system remain consistent across periods when the system is culled. The essential steps are identifying those properties that are important, and seeking the extent to which uncertain knowledge of initial state influences the ability to predict future state. While the examples are specific to particular systems, the properties of these systems on which we rely – weak influence of initial state, state based behavior, and a controller producing perturbed constant motion – are all properties common to many systems we wish to model.

While the models used here were hand generated, the existence of general solutions for commonly occurring systems suggests that much of the work required may be automated, possibly through the application of existing statistical methods for estimating the influence of initial conditions[10] and modeling distributions that appropriately reflect the behavior of systems over time. The ideal analysis tool would take a detailed, possibly annotated, dynamic model of the system and provide a new model to support culling in a consistent and high performance virtual environment.

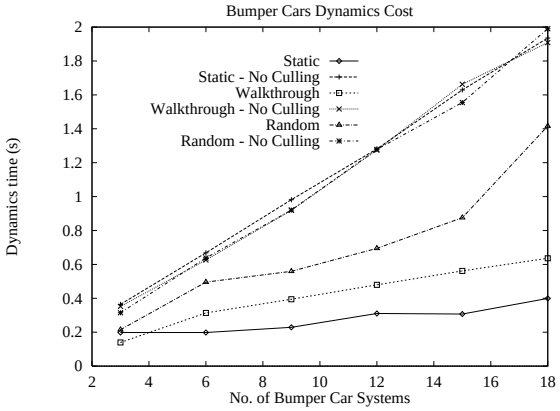

Figure 3: *Mean real time spent on dynamics per second of virtual time plotted against the number of bumper car systems, under a variety of viewpoint movements, comparing culled dynamics with unculled dynamics. Views are as per figure 2. As the number of cars increases, the speedup obtained by culling dynamics for a static view increases too, because the number of objects that are never rendered increases. The speedup is smaller for the slowly moving view because occasionally an object comes into view, and its state must be advanced to produce data for the current frame; however, the speedup again increases with the number of cars, because the number of invisible objects is still increasing. Viewpoints that move quickly, like the chaotic view, do not allow much speedup, although the speedup grows with the number of cars in this case, too; the speedup is better than the case of the Tilt-A-Whirl because our state generation strategies are more efficient for this case.*

References

[1] Arnol'd, V. I. *Mathematical Methods of Classical Mechanics*. Graduate texts in mathematics. Springer-Verlag, New York, 1989.

[2] Funkhouser, T. A., Sequin, C. H., and Teller, S. J. Management of large amounts of data in interactive building walkthroughs. In *Proceedings 1992 Symposium on Interactive 3D Graphics* (29 March - 1 April 1992), pp. 11–20. Cambridge, Massachusetts.

[3] Gottschalk, S., Lin, M. C., and Manocha, D. Obbtree: A hierarchical structure for rapid interference detection. In *Computer Graphics* (1996), ACM SIGGRAPH, pp. 171–180. New Orleans, LA.

[4] Hahn, J. K. Realistic animation of rigid bodies. In *Computer Graphics* (August 1988), vol. 22(4), ACM SIGGRAPH, pp. 299–308. Atlanta, Georgia.

[5] Hodgins, J. K., and Carlson, D. A. Simulation levels of detail for real-time animation. Tech. Rep. GIT-GVU-96-32, Graphics, Visualization and Usability Center, Georgia Institute of Technology, 1996.

[6] Kautz, R. L., and Huggard, B. M. Chaos at the amusement park: Dynamics of a tilt-a-whirl. *American Journal of Physics 62*, 1 (January 1994), 59–66.

[7] Naylor, B., Amanatides, J., and Thibault, W. Merging bsp trees yields polyhedral set operations. In *Proceedings of SIGGRAPH 90*, Computer Graphics, 24, 4 (August 1990), pp. 115–124. Dallas, Texas, August 6-10, 1990.

[8] Pajon, J., et al. Building and exploiting levels of detail: An overview and some vrml experiments. In *VRML '95 First Annual Symposium on the Virtual Reality Modeling Language* (Dec 14-15 1995), pp. 117–122. San Diego, California.

[9] Setas, M. N., Gomes, M. R., and Rebordão, J. M. Dynamic simulation of natural environments in virtual reality. In *SIVE95: The First Workshop on Simulation and Interaction in Virtual Environments* (July 1995). University of Iowa, Iowa City, IA.

[10] Wolf, A., and Bessoir, T. Diagnosing chaos in the space circle. *Physica D 50* (1991), 239–258.