MULTI-BODY SIMULATION IN VIRTUAL ENVIRONMENTS

Mashhuda Glencross and Alan Murta
Department of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL
United Kingdom

Email glencross@cs.man.ac.uk

Abstract

This paper describes a simulator which uses a unique hybrid particle system and dynamic constraint based model to compute the behaviour of interacting multi-body systems in a virtual environment. Simulations can be influenced by the user at run-time, so there are important interaction issues addressed. The need for such systems in virtual reality applications ranging from tissue modelling in surgical training software to use in entertainment, is apparent. Purely, a visual effect does not suffice in this context, as the behaviour of objects in the scene must be plausible enough to convince the user. Our approach provides a unified system for simulating a wide spectrum of deformable and/or rigid material behaviours in virtual environments. An example of a virtual chain, which the user can interact with, is provided.

Keywords: constraint based, physically based, interaction forces, rigid bodies, multi bodies, particle systems, virtual reality.

1 Introduction

To simulate the behaviour of deformable or rigid materials subject to forces requires a physically based model. Many such models have been successfully used in computer animation. Constraint based modelling is a technique in which forces are calculated in order to control the motion of a system. It is frequently used to animate articulated rigid body systems [2, 1, 7, 5, 9]. Other approaches concentrate purely on animating deformable materials [8, 10, 16, 17]. However, most of these approaches are too computationally intensive to be used in virtual reality (VR) applications, where a real time response is of paramount importance.

Ideally we wish to simulate both rigid and deformable material responses, but this is complex and some VR systems which have evolved use purely rigid body models. Other systems use spring mesh primitives to build up larger bodies, such as Holton's muscle model [6]. Such models are confined to deformable material behaviour only. A unified model capable of simulating both rigid and deformable behaviours is very appealing because the real world has both. Chapman and Wills' unified model uses modal analysis [4] to simplify the finite element equations of rigid and deformable motion. However, it is difficult to produce a desired behaviour using finite element based techniques because all elements are treated in the same way, and the parameters which govern their deformation behaviour are unintuitive.

Our approach is to use a hybrid model based on particle primitives which can be connected by rigid connections or force functions of any type. The user can simulate the desired behaviour by specifying complex interactions between particles and incorporating rigid and/or articulating components. This produces very interesting material behaviours.

We also address some important user interaction issues arising from the highly interactive nature of VR systems. Our scenes are very dynamic and the user may make or break model primitives at any point during the course of the simulation. User interaction is impossible to predict and so we take certain measures which are described in sections 2.3 and 4 to enable the model to deal with this.

2 Model Primitives

We provide a number of model primitives (described in section 2.1) and a high level library of force functions. This enables us to build up interesting materials with any combination of rigid and deformable characteristics. Figure 1 shows an example of a material which could be simulated using rigid building blocks and articulat-

ing hinges. The non-auxetic material becomes thinner when a deforming force is applied, whereas the auxetic material fills out [3].

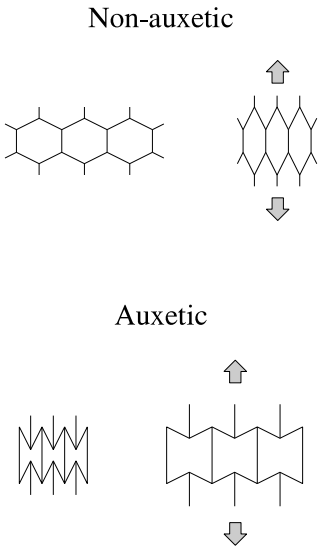

Figure 1: Non-auxetic and auxetic material deformation behaviours.

It is also important to note that any force function can be programmed in a script which defines the scene if a function from the library does not suffice. This leads to a great deal of flexibility when assigning behaviours to bodies.

2.1 Interacting Particles

Researchers have used particle based models for simulating a wide range of natural phenomena from fire and waterfalls to wind-blown leaves [11, 14, 18]. These effects are achieved by rendering particles in different ways and changing their attributes as some function of time. All of these particle based models have some type of force acting on them. Typically these are global forces such as gravity and act on all the particles in the scene indiscriminately [11]. More complex effects such as flocking can be achieved by introducing particle-particle interactions [12].

We use a unique representation of particles as being composed of one or more points, depending on whether or not a particle is also a hinge. A point is merely a point mass in space with some data associated, for example colour. The benefits of this approach are described in further detail in section 2.3.

2.1.1 Families of Particles

Particles can be considered at a high level as belonging to families, where a family is a group of one or more particles. Their interaction behaviour can be specified by non-rigid connections (force functions) which apply either within the family and/or between it and other families.

Families of particles may have multiple force functions attached due to the flexibility inherent in the model. This can lead to some very sophisticated cumulative behaviour. For example, the hydrophobic type behaviour shown in figure 2.

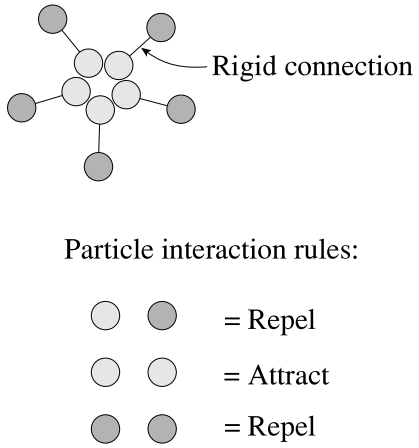

Figure 2: A system of bodies with complex interactions.

There are five bodies shown in the scene, each comprises two particles connected by a rigid component. The particles are from different families and their relative interaction rules are shown. In three dimensions this type of behaviour can produce surface forming systems [15] but at comparatively little computational cost.

2.2 Rigid Bodies

Much of the research into rigid body systems has focused on simulating the motion of purely rigid bodies, subject to forces such as gravity for animation purposes. A simple rigid body's motion is affected most significantly by the distribution of mass within it. There is always one unique point called the centre of mass which can be used to compute the motion of such a body. We provide such bodies which may also have interaction behaviours. These types of bodies can interact with anything in the scene if a behaviour is specified, but individual particles within any given body are subject to a geometric rigidity constraint. Thus the relative positions of such particles within a body remains unchanged.

If the body contains joints capable of articulating the problem of computing its motion is not so simple.

2.3 Hinge Particles

When a particle is also a hinge our representation of particles becomes important. Notionally, a hinge particle behaves just like an ordinary particle with associated characteristics such as mass, colour, lifetime etc. However, it is made up of two or more member points which belong to the bodies it relates. Point data is stored in a ring data structure, as shown in figure 3. Each member point has its own local data which declares its properties, a pointer to the next member and access to some global information about the hinge particle itself.

This representation enables intuitive access to members when applying hinge constraints. Furthermore, it provides some consistent framework for assigning and applying the hinge particles properties when two or more bodies with different characteristics are connected together.

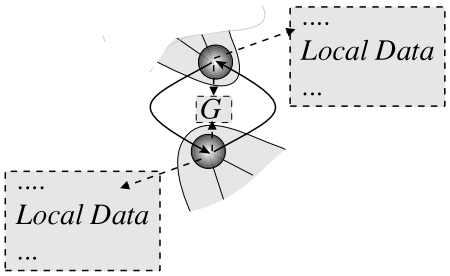

Figure 3: Figure showing two points in a ring making up a particle, each with its own 'Local Data', and sharing 'Global Data'

Enhancing the simulator to respect the sharing of particles between bodies and so provide 'hinging' adds a great deal of complexity to the model described. To keep the hinges intact we use one from our suite of dynamic constraints which is similar to Barzel and Barr's point to point constraint [1, 2]. However, our technique is more suited to VR applications where performance, and the need to change the mass of bodies at run-time is of particular importance.

Barzel and Barr retain bodies in their own co-ordinate frame, transforming them into world space as required. This has the advantage that the moment of inertia remains constant. However, this assumes that the scene is relatively static, in that the mass of bodies remains unchanged during a simulation.

We argue that Barzel and Barr's coordinate frames approach can be traded off against the need to compute the inertia matrix for bodies at the start of each frame. Recomputing the inertia matrix every frame is necessary in our approach but it gives us mass and topology changes without any further computational cost. A body's mass is dynamic in our system, the user may combine two bodies to make one articulating body or may pull one apart. In either case, the mass of the resulting body or bodies will probably be different. Figure 4 shows two bodies which are scripted to combine and make a hinge point when the hinge constraint is met at point P.

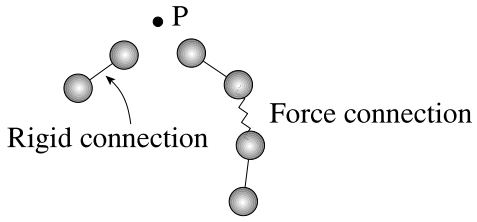

Figure 4: Two bodies approaching the constraint point P

2.4 Structural Primitives

We provide a high level library of structural primitives which are the building blocks of a scene. The configuration of these arise from molecular data and regular polyhedra. The main reason for providing this is to enable easy description of very complex scenes, this approach gives the user significant control over the configuration of the stable state of the system. The user has a choice about the type of connections to specify between particles in these structures, rigid or deformable. In addition routines are provided to build these structures from rigid bodies and articulating joints.

3 Performance Considerations

Articulates are a concept provided purely to improve the performance of the system. Articulates are groups of bodies related to each other by hinge particles, and are hidden from the user. This grouping of bodies reduces the problem of finding unknown forces to the smallest number of particles possible, allowing the solver to only work within the domain of one articulate at a time. This concept also potentially lends itself to a parallel algorithm in which articulates, which need to be solved, can be dispatched to free processors.

4 Temporal Sequencing

Temporal sequencing is provided to choreograph a simulation. For example, at some time the user may want all the connections in certain bodies to break and so effect a melting behaviour. We can create or destroy

connections as a function of time or system 'temperature'. In addition it is possible to do the same to bodies and hinges. This facility poses a problem at run-time as it may be defined in a script which has no knowledge of the user's intervention. Hence, run-time conflicts may occur between the script and the actual scene at any time t. For example, we may have scripted all the connections in a body *body*1 to break at $t = 100$, however, the user may have destroyed *body*1 at $t = 50$. In this particular case we would not apply the event.

5 The Simulator

Computing the motion of primitives for each frame is illustrated by the following pseudo-code:

1. Apply forces associated with non-rigid connections to particles.
2. For each Articulate:
   (a) Compute Centre of mass, Mass, Moment of Inertia, Net Force/Torque for each body in Articulate ; Tally number of unknown Forces in Articulate.
   (b) If number of unknown forces > 0
      i. Call Solver to find unknowns
   (c) Apply forces to get new positions of all particles in an Articulate.
3. Draw scene.

The solver calls a method which writes the unknowns into the particles which they correspond to, then recomputes accelerations and velocities for each body. The constrained points are then revisited and a metric which should become zero is calculated. For example, "how far is a constrained point from its target destination?" These values are finally accumulated into a vector for returning as the results passed out.

The solver uses a Newton Raphson iteration with line backtracking and the time frame is advanced by performing a simple Euler integration of all the forces. The simulator engine is written in C++ and interfaced to Perl which is used as the system scripting language.

6 Results

We give a simple example of a virtual chain falling under gravity, Figure 5 shows the chain at two instances during its motion. This example was chosen to give a comparison with Barr et al. The chain is composed of rigid bodies and hinge points. It is driven by the user in virtual space. We achieve this by constraining a particle on the first body in the chain to follow the mouse. Any particle can be constrained in this way, in fact in a simulation with two mice, this type of chain can be manipulated via two particles. The user can move the chain about and pull it apart or play with it as if it were a virtual skipping rope.

Figure 5: A virtual chain.

7 Conclusions

The contribution of this work lies in taking aspects from particle based models and rigid body systems and combining them to create a unified approach to simulating plausible material behaviour in virtual environments. Our system enables us to build very sophisticated scenes and set up complex interactions between primitives. In addition, the user can interact with the simulation and affect its course. We have also touched upon some very interesting issues which arise from using scripting in a highly interactive system.

A Appendix

A.1 Equations of motion

The equations of motion used in the work described in this paper are derived from the standard Newtonian equations of motion. To evaluate our constraint forces we start with the definition of absolute velocity as the rate of change of a particle's position vector $\vec{p}$, written as

$$\vec{v} = \frac{d\vec{p}}{dt} = \dot{\vec{p}} \tag{1}$$

In turn, the rate of change of a particle's velocity is defined to be the particle's acceleration.

$$\vec{a} = \frac{d\vec{v}}{dt} = \dot{\vec{v}} = \frac{d^2\vec{p}}{dt^2} = \ddot{\vec{p}} \tag{2}$$

Since $\vec{a} = \vec{f}/m$ gives the instantaneous acceleration of a particle, it is necessary to integrate Equation 2 to obtain the path of a particle.

$$\vec{p} = \int \vec{v}\, dt + \vec{p'} = \int\int \vec{a}\, dt^2 + \vec{p'} \tag{3}$$

where $\vec{p'}$ derives from the boundary condition.

Similarly, rotational torques for bodies are calculated by summing cross products of forces and particle positions as in Equation 4. This provides an angular acceleration which is also integrated. The resulting vector is converted to a matrix [13] and applied to each particle in the body. This vector to matrix conversion is performed by a function $\mathcal{A}$.

$$\vec{e} = \vec{p} \wedge \vec{f} \tag{4}$$

Equation 5 specifies the motion of bodies in the simulator. 5a moves the body to the origin and performs a rotation, while 5b moves it back. Then 5c performs an appropriate translation.

$$\begin{aligned} \vec{p'}_{A.n} &= \mathcal{A}\left(\vec{V}_A += c\vec{I}^{-1} \sum_{n=1}^{N} \vec{f}_{A.n} \wedge (\vec{p}_{A.n} - \vec{P}_A)\right) \\ &\quad .(\vec{p}_{A.n} - \vec{P}_A) && \text{(5a)} \\ &+ \vec{P_A} && \text{(5b)} \\ &+ (\vec{V}_A += \vec{F}_A M_A^{-1}) && \text{(5c)} \end{aligned}$$

Our convention to represent a body is a capitalised subscript; A in the above case. The lower case subscript indicates a particle which belongs to the body.

Acknowledgements

I would like to thank my husband Nick Glencross for proof reading and constructive criticism. I would also like to thank Dr. Richard Zobel for his encouragement.

References

[1] Ronen Barzel. *Physically-Based Modelling for Computer Graphics.* Academic Press, San Diego, CA, 1992.

[2] Ronen Barzel and Alan Barr. A modelling system based on dynamic constraints. *Computer Graphics*, 22(4):179–188, August 1988.

[3] Maria Burke. A stretch of the imagination. *New Scientist*, 154(2085):36–39, June 1997.

[4] Peter M. Chapman and Derek P.M. Wills. Towards a unified physical model for virtual environments. In *Proc. 4th UK VR-SIG Conference*, Brunel University, UK., November 1997.

[5] Jean-Dominique Gascuel. Displacement constraints: a new method for interactive dynamic animation of articulated solids. In *Third Eurographics Workshop on Animation and Simulation*, Cambridge, England, September 1992.

[6] Matthew Holton and Simon Alexander. Soft cellular modelling: A technique for the simulation of non-rigid materials. In *Computer Graphics: Developments in Virtual Environments*, pages 449 – 460. Academic Press, 1995.

[7] Devendra Kalra and Alan H. Barr. A unified framework for constraint-based modeling. In *Proc. CG International '92*, pages 675–695, 1992.

[8] G. Miller and A. Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 1989.

[9] C. Van Overveld. An iterative approach to dynamic simulation of 3-d rigid-body motions for real-time interactive computer animation. *The Visual Computer*, 7:29–38, 1991.

[10] J.C. Platt and A.H. Barr. Constraint methods for flexible models. *Computer Graphics*, 22(4):279–288, August 1988.

[11] W.T. Reeves. Particle systems – a technique for modeling a class of fuzzy objects. *Computer Graphics*, 17(3):359–376, July 1983.

[12] C.W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, July 1987.

[13] F. P. Sayer and J. A. Bones. *Applied Mechanics A modern approach.* Chapman and Hall, London, 1990.

[14] Karl Simms. Particle animation and rendering using data parallel computation. *Computer Graphics*, 24(4):405–413, August 1990.

[15] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. Technical Report CRL 91/14, Cambridge Research Lab, December 1991.

[16] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, December 1988.

[17] D. Terzopoulos, J. Platt, and K. Fleischer. From goop to glop: Heating and melting deformable models. In *Graphics Interface*, pages 219–226, June 1989.

[18] Jakub Wejchert and David Haumann. Animation aerodynamics. *Computer Graphics*, 25(4):19–22, 1991.