Panel Session

Physically-Based Modeling: Past, Present, and Future

Co-Chairs: Demetri Terzopoulos, *Schlumberger Laboratory for Computer Science*
John Platt, *Synaptics*

Speakers: Alan Barr, *California Institute of Technology*
David Zeltzer, *MIT Media Lab*
Andrew Witkin, *Carnegie Mellon University*
Jim Blinn, *California Institute of Technology*

My name is Demetri Terzopoulos and my co-chair, John Platt, and I would like to welcome you to the panel on Physically-Based Modeling -- Past, Present and Future. I'll start by introducing the panelists; the affiliations you see listed on the screen are somewhat out of date.

I'm Program Leader of modeling and simulation at the Schlumberger Laboratory for Computer Science in Austin, Texas, and I was formerly at Schlumberger Palo Alto Research. I'll speak on the subject of deformable models.

John Platt, formerly of Cal Tech, is now Principal Scientist at Synaptics in San Jose, California. He will be concentrating on constraints and control.

Alan Barr is Assistant Professor of computer science at Cal Tech. Last year he received the computer graphics achievement award. He'll speak about teleological modeling.

David Zeltzer is Associate Professor of computer graphics at the MIT Media Laboratory. He will be speaking on interactive micro worlds.

Andrew Witkin, formerly of Schlumberger Palo Alto Research, is now Associate Professor of computer science at Carnegie Mellon University. He will speak about interactive dynamics.

Last but not least, we have with us James Blinn, who of course needs no introduction. Formerly of JPL, he is now Associate Director of the Mathematics Project at Cal Tech. He says he'll have several random comments to make against physically-based modeling.

I was also asked by the SIGGRAPH organizers to remind the audience that audio and video tape recording of this panel is not permitted.

Many of you are already familiar with physically-based modeling, so I will attempt only a very simple introduction to this, in my opinion, very exciting paradigm. Physically-based techniques facilitate the creation of models capable of automatically synthesizing complex shapes and realistic motions that were, until recently, attainable only by skilled animators, if at all. Physically-based modeling adds new levels of representation to graphics objects. In addition to geometry -- forces, torques, velocities, accelerations, kinetic and potential energies, heat, and other physical quantities are used to control the creation and evolution of models. Simulated physical laws govern model behavior, and animators can guide their models using physically-based control systems. Physically-based models are responsive to one another and to the simulated physical worlds that they inhabit.

We will review some past accomplishments in physically-based modeling, look at what we are doing at present, and speculate about what may happen in the near future. The best way to get a feel for physically-based modeling is through animation, so we will be showing you lots of animation as we go along.

I would like to talk about deformable models, which are physically-based models of nonrigid objects. I have worked on deformable models for graphics applications primarily with Kurt Fleischer and also with John Platt and Andy Witkin. Deformable models are based on the continuum mechanics of flexible materials. Using deformable models, we can model the shapes of flexible objects like cloth, plasticine, and skin, as well as their motions through space under the action of forces and subject to constraints.

Please roll my Betacam tape. Here is an early example of deformable surfaces which are being dragged by invisible forces through an invisible viscous fluid. Next we see a carpet falling in gravity. It collides with two impenetrable geometric obstacles, a sphere and a cylinder, and must deform around them. The next clip shows another elastic model. It behaves like a cloth curtain that is suspended at the upper corners, then released.

Here is a simulated physical world -- a very simple world consisting of a room with walls and a floor. A spherical obstacle rests in the middle of the floor. You're seeing the collision of an elastically deformable solid with the sphere. Of course, we're also simulating gravity.

We've developed inelastic models, such as the one you see here which behaves like plasticine. When the model collides with the sphere, there's a permanent deformation. By changing a physical parameter, we obtain a fragile deformable model such as the one here. This deformable solid breaks into pieces when it hits the obstacle.

Deformable models can be computed efficiently in parallel. This massively parallel simulation of a solid shattering over a sphere was computed on a connection machine at Thinking Machines, with the help of Carl Feynman.

Here is a cloth-like mesh capable of tearing. We're applying shear forces to tear the mesh. The sound you're hearing has been generated by an audio synthesizer which was programmed by Tony Crossley so that it may be driven by the physical simulation of the deformable model. Whenever a fiber breaks, the synthesizer makes a pop. Keep watching the cloth; we get pretty vicious with it.

Deformable models are obviously useful in computer graphics, but they are also useful for doing inverse graphics; that is to say, computer vision.

For example, here we see an image of a garden variety squash. Using a deformable tube model, we can reconstruct a three dimensional model of the squash from its image, as shown. Once we have reconstructed the model from the image, we can

rotate the model to view it from all sides. You can see, we have captured a fully three dimensional model from that single, monocular image. That's a basic goal of computer vision.

Kurt Fleischer, Andy Witkin, Michael Kass, and I used this deformable model based vision technique to create an animation called *Cooking with Kurt*. We wanted to mix live video and physically-based animation in this production. You see Kurt entering a kitchen carrying three vegetables. We captured deformable squash models from a single video frame of the real squashes sitting on the table -- this particular scene right here. Now the reconstructed models are being animated using physically-based techniques. The models behave like very primitive actors; they have simple control mechanisms in them that make them hop, maintain their balance, and follow choreographed paths. The collisions and other interactions that you see are computed automatically through the physical laws, and they look quite realistic. It's difficult to do this sort of thing by hand, even if you're a skilled animator.

This second tape will show you some of the physically-based modeling we're up to now at the Schlumberger Laboratory for Computer Science. Keith Waters and I are working on interactive deformable models. We're now able to compute and render deformable models in real time on our Silicon Graphics Iris 240 GTX computer. For example, here is a simulation of a nonlinear membrane constrained at the four corners and released in a gravitational field. Watch it bounce and wiggle around.

Here you're seeing a physically-based model of flesh. It's a three dimensional lattice of masses and springs with muscles running through it. Again, this is computed and displayed in real time. You can see the muscles underneath displayed as red lines. They're fixed in space at one end and attached to certain nodes of the lattice model at the other end. By contracting the muscles we can produce deformations in this slab of -- whale blubber, if you will. We did this simulation as an initial step towards animating faces using deformable models as models of facial tissue. And of course, the muscle models make good facial muscles.

The next clip will demonstrate real time, physically-based facial animation on our SGI computer. Here we see the lattice structure of the face. Let's not display all of the internal nodes so that we can see the epidermis of the lattice more clearly. There. Now we're contracting the zygomatic muscle attached to one edge of the mouth -- now both zygomatics are contracting to create a smile. The muscles inside the face model are producing forces which deform the flesh to create facial expressions.

Now the epidermis polygons are displayed with flat shading. Next we contract the brow muscles. Here the epidermis is being shaded smoothly. Finally, we relax the muscles and the face returns to normal.

An important reason for applying the physically-based modeling approach to facial animation is realism. For instance, the facial tissue model automatically produces physically realistic phenomena such as the laugh lines around the mouth and the cheek bulges that you see here.

Keith videotaped this animation off of our machine only last week. Our next step will be to develop control processes to coordinate the muscles so that the face model can create a wide range of expressions in response to simple commands. Keith's prior work on facial animation, published in SIGGRAPH 87, showed how one can go about doing this using muscle model processes. Beyond muscle control processes, we're also interested in incorporating vocoder models -- that is, physically-based speech coding and generation models, so that this face can talk to you.

The tape will end soon, so I'll release the podium to Dr. John Platt, who will talk about constraint methods and control. Thank you.

John Platt
Synaptics

Hello. I'm John Platt and I'm going to tell you one major idea that I have found to be very useful in working with physically-based models. Animation is simulation plus control. I worked on this idea with Al Barr while I was a graduate student at Cal Tech.

I claim there are two necessary ingredients to make interesting animation. One of them is the physical simulation of elasticity. Demetri talked about this a little bit. You need to have models that obey the theory of elasticity. In other words, you use Newton's laws to make the models act naturally. The animation looks natural, because the theory of elasticity describes the way flexible models actually behave.

Physical simulation is also nice because it's automatic. If you have a simulator that simulates an elastic object, it can have hundreds of variables. Trying to do key framing would be very difficult: you would have to specify hundreds of splines in order to make the animation.

In addition to the physical simulation, elastic models need to be controlled. Models should follow basic rules which create good animation. For example, you usually don't want models to fly through each other -- unless you want that particular effect in your animation. Objects should bounce off each other. They should be able to be incompressible or moldable.

More generally, you want to guide models. You don't want just a pure simulation. You want to be able to specify some amount of control and then let the rest be automated. So you can specify a few degrees of freedom and leave the other few hundred to the computer. So I claim this means you want to have both simulation plus control to make animation.

Let me show you some examples of animation made using constrained flexible models that will illustrate this principle.

What you're first going to see is an elastic trampoline with a sphere above it. With constraints, I specify that this sphere should not penetrate the trampoline. And you see, it doesn't. It bounces; it stays above the trampoline.

In the next example, I use constraints to try to assemble complex objects out of simple objects, and I also use constraints to position the objects where I wanted. Here, I specify a few constraints and the system automatically positions the models to create a double trampoline.

You don't have to confine yourself to surfaces. Very interesting animations result when you simulate elastic solids. So here I'm going to make a jello cube. Now, I pick up the jello cube with constraints. Gravity is applied to the jello cube so that it falls. The grey table is made by constraints: I'm constraining the jello cube to stay above the table.

Finally, you can make reasonably complex animations involving hundreds of variables. This is an example of such an animation using both flexible models and constraints.

This animation was made with the help of a lot of my friends from Cal Tech while I was there, and in fact, we did it up at Apple on their Cray. So I'd like to thank all those people.

In conclusion, I want to reiterate: if you want to make very complex and interesting animation, then I think you need both simulation and control. The simulation can be any sort of physics. It doesn't have to be elasticity; it could be fluid mechanics or neutrino physics or whatever. But you need both simulation and control to create animation that does what you want.

I'm going to pass the speakership on to Professor Al Barr from Cal Tech.

Alan Barr
California Institute of Technology

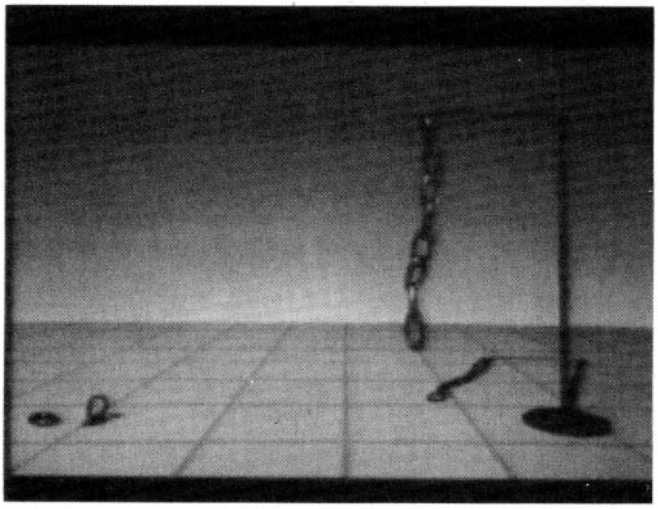
We're talking here about physically-based modeling and an obvious question is: Well, gee, physics has been around for a few hundred years -- don't people in computer graphics know freshman physics? Why did it take so long for these people to use physics in their work?

The answer is that physics by itself does what it wants to do -- It doesn't want to do what you want to do.

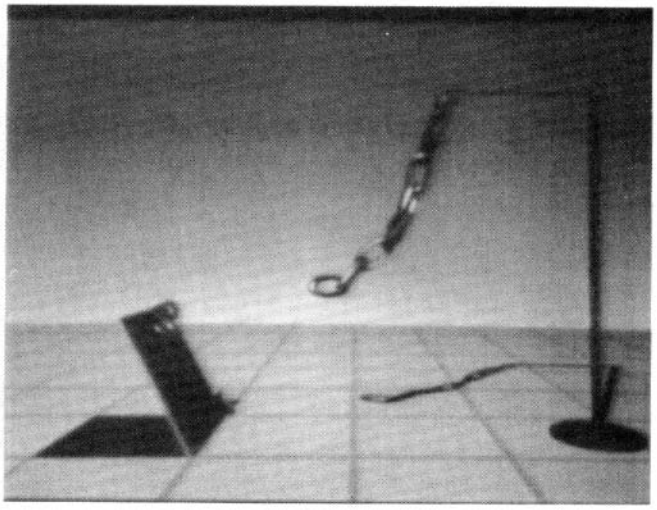
In terms of the scientific foundations of computer graphics, the world view of what I'm talking about is that modeling is making mathematical abstractions of objects, and that rendering is making pictures. My prediction is that there is going to be a large and increasing role in science for the modeling that we're doing in computer graphics. After all, in science what you're trying to do is to make a predictive model that agrees with experiment. Since so much of what is done in science is modeling, the techniques that we're demonstrating today will make it possible to do scientific modeling much more easily than it can be done at present.

For example, let's say I wanted an elastic model that is isotropic -- the same in all directions. It has constraints in that it does not pass through this object and it does not pass through that object, and interacts with rigid and flexible bodies. Now that's a very compact description of the model. How long would it take us to actually program that up? It takes us quite some time. So, with advanced modeling tools in which those properties that I've just described are primitives, we'll be able to do a lot more modeling in a shorter period of time, and the whole world will be a better place.

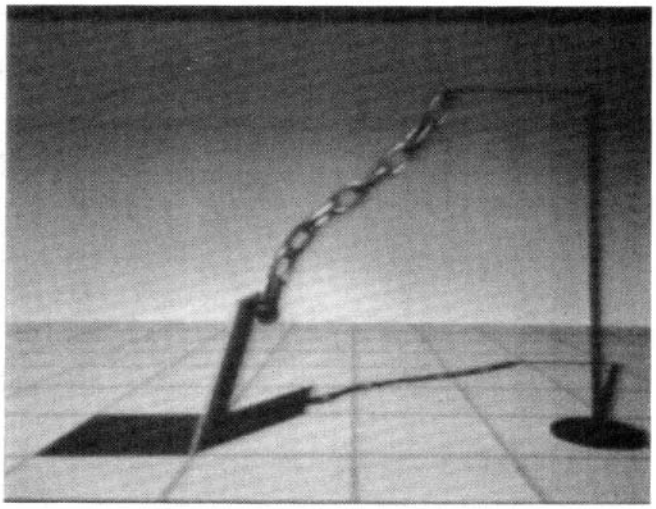
Basically, in the modeling process you abstract away the features you wish to model and you represent them. Then you implement those features. I use the abstraction that a teleological object takes goals and an incomplete specification of an object, and produces a complete geometric description.



— BARR - SLIDE 1 —



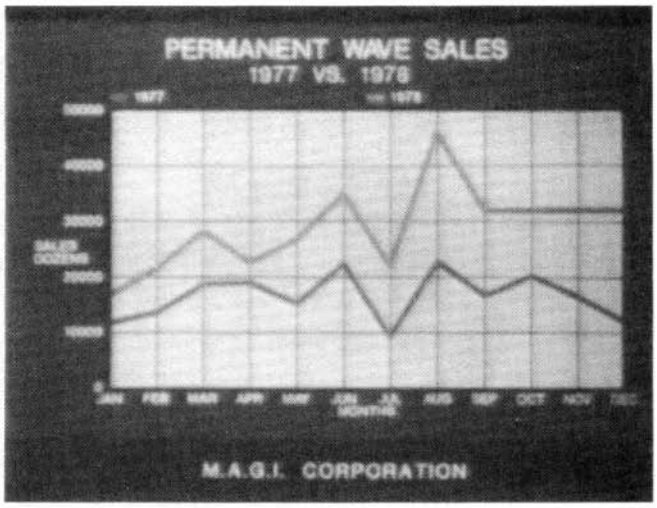
— BARR - SLIDE 2 —



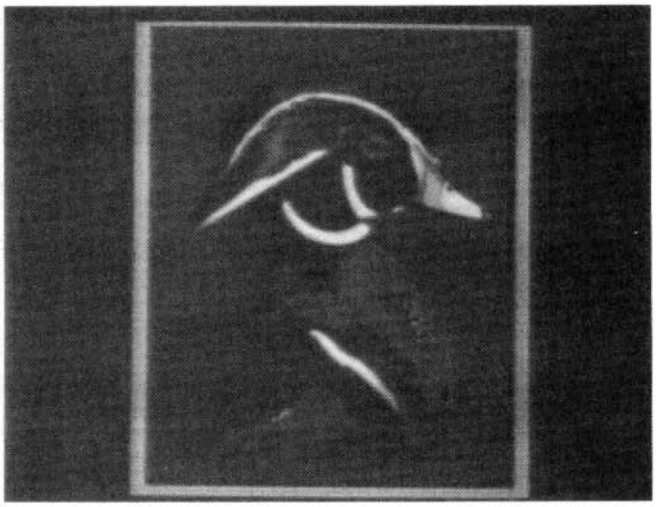
— BARR - SLIDE 3 —

For instance, here I have a chain and I should be able to ask the bottom link of the chain to hook to the trapdoor lid. It would be very nice if it would just do it.

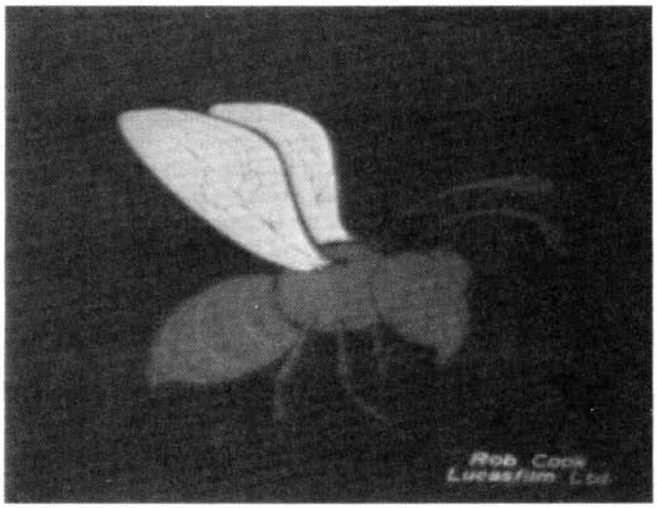
Teleological methods, such as constraint methods, deal with the forces and with the constraints simultaneously. Basically, the abstraction of the objects consists of both the goals of behavior and the physics. In one framework, you have geometric constraint properties, mechanical properties, the control of your objects, and the parameters that describe the sizes of your objects.



— BARR - SLIDE 4 —

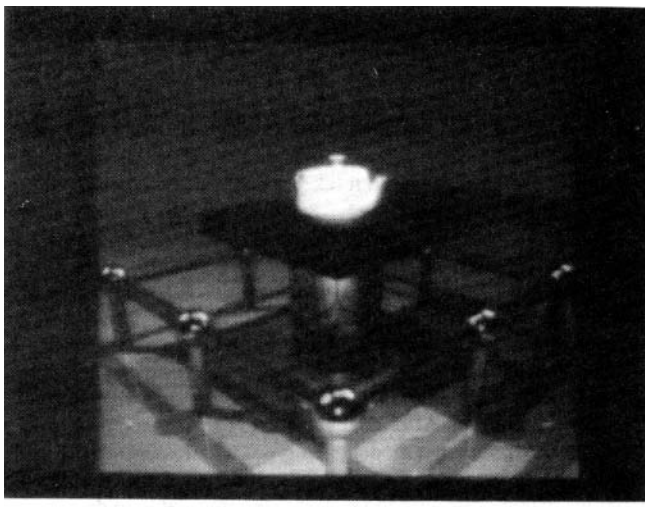


— BARR - SLIDE 5 —

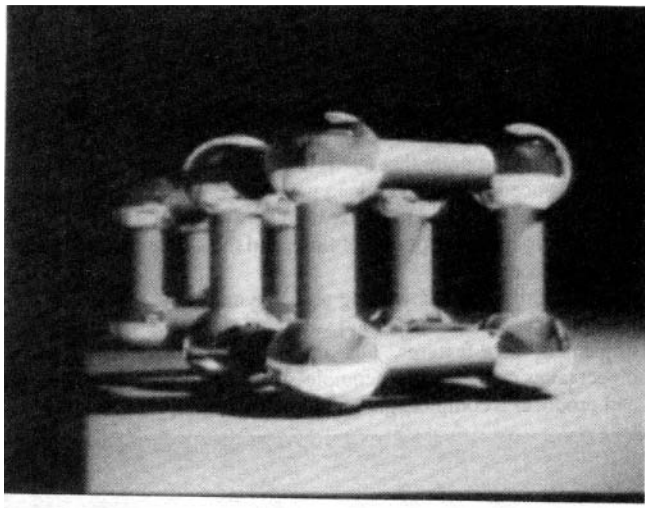


— BARR - SLIDE 6 —

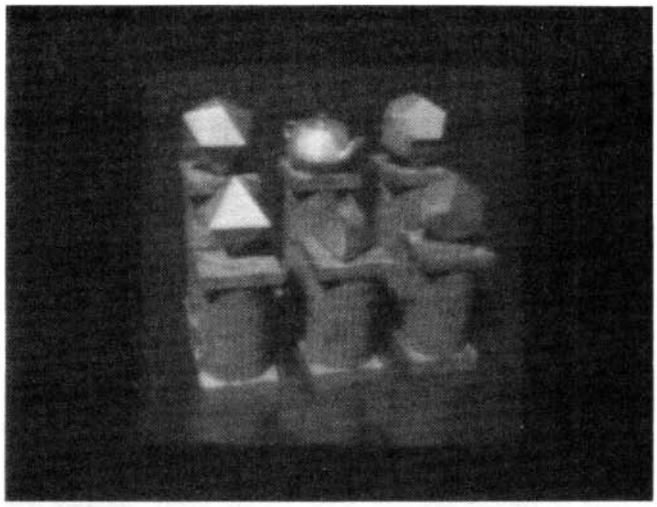
The simplest level of abstraction of an object is an image. The next level of abstraction is that an object is a shape.



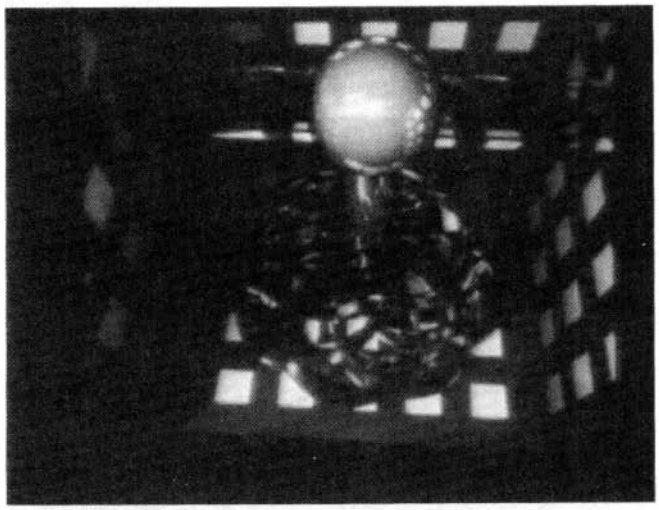
— BARR - SLIDE 7 —



— BARR - SLIDE 8 —



— BARR - SLIDE 9 —

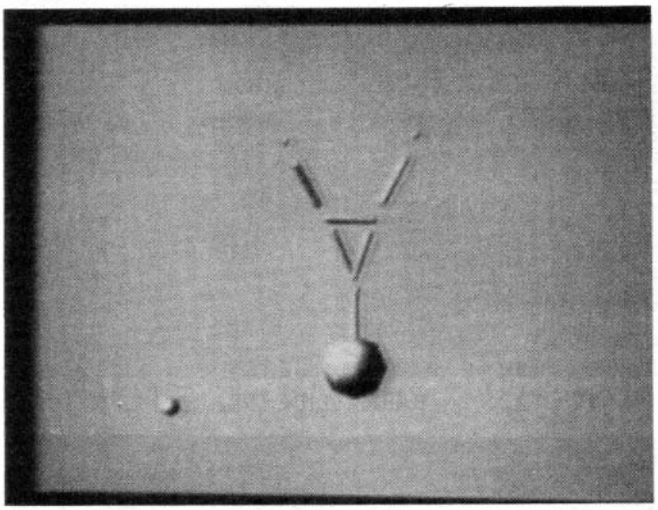


— BARR - SLIDE 10 —

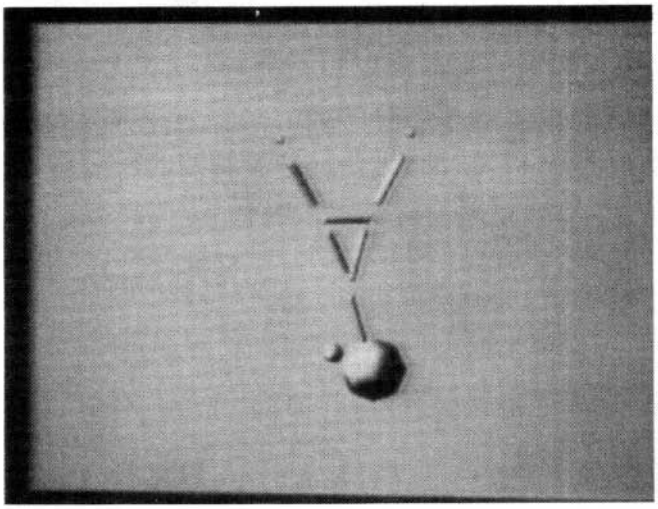
— BARR - SLIDE 11 —

— BARR - SLIDE 12 —

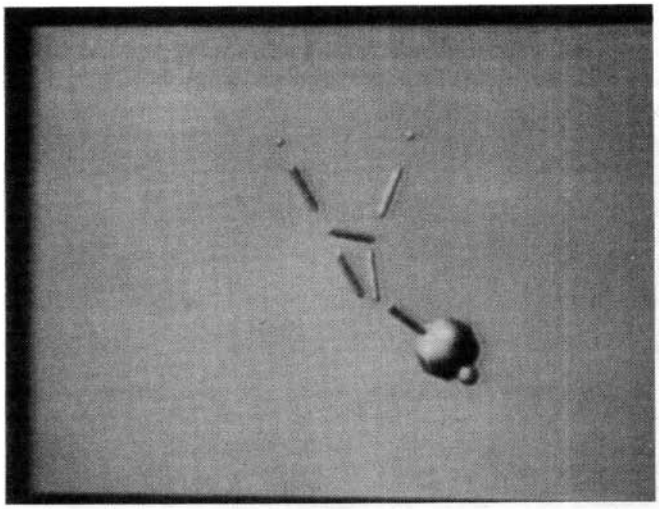
So here are objects that are described strictly geometrically: no physics. This is a picture by Dave Kirk and Jim Arvo and they claim that since the Greeks knew about polyhedra, that a new platonic solid has been found. It's the middle object in the back.



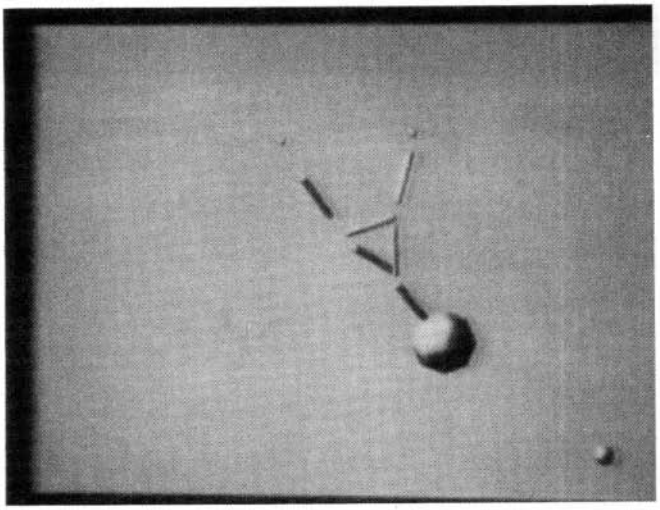
— BARR - SLIDE 13 —



— BARR - SLIDE 14 —

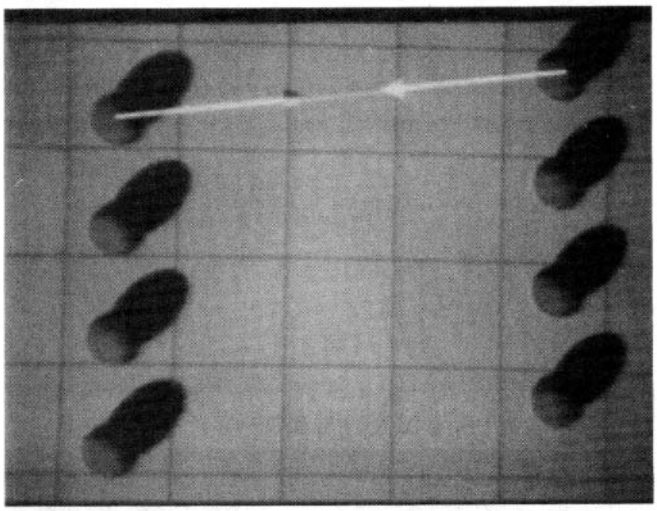


— BARR - SLIDE 15 —

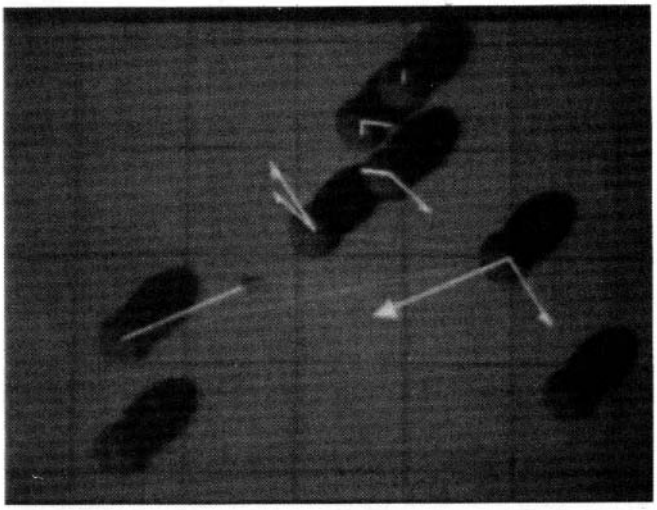


— BARR - SLIDE 16 —

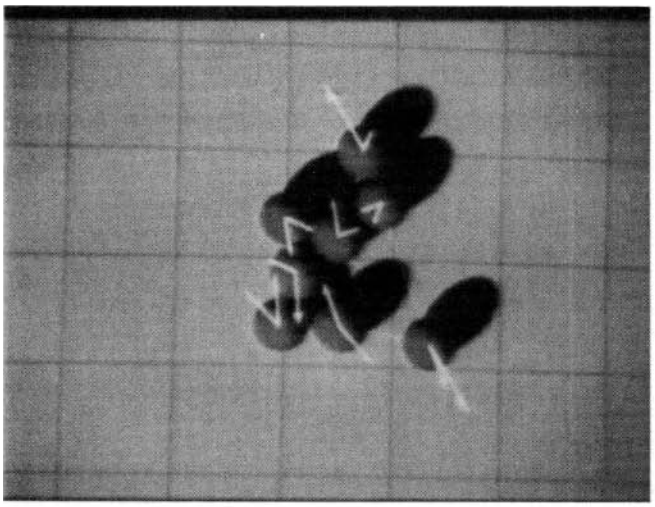
The next level of abstraction is physics. An object is its physical behavior, but you can see that physics alone doesn't necessarily have constraints.



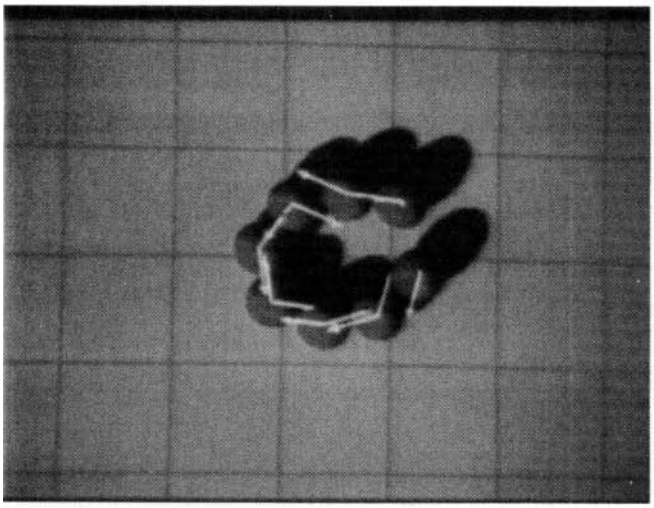
— BARR - SLIDE 17 —



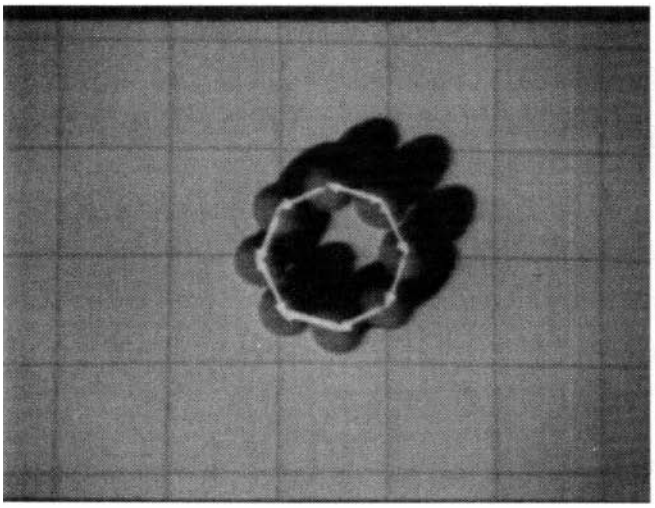
— BARR - SLIDE 18 —



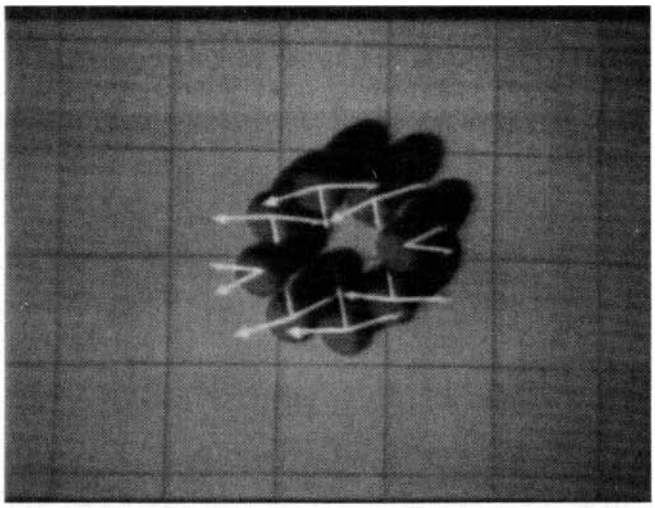
— BARR - SLIDE 19 —



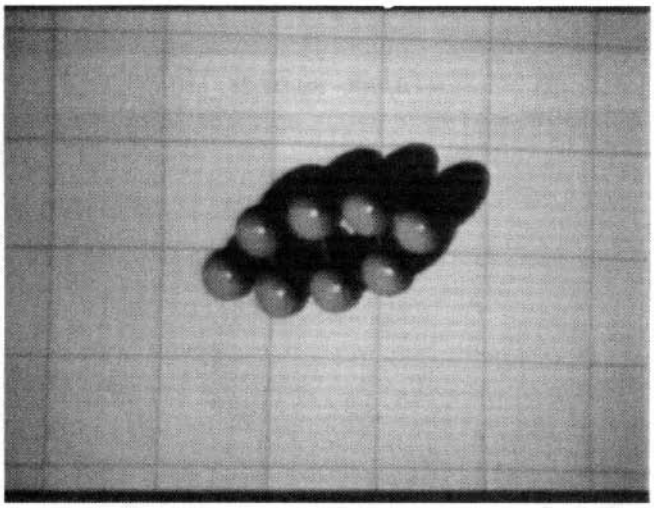
— BARR - SLIDE 20 —



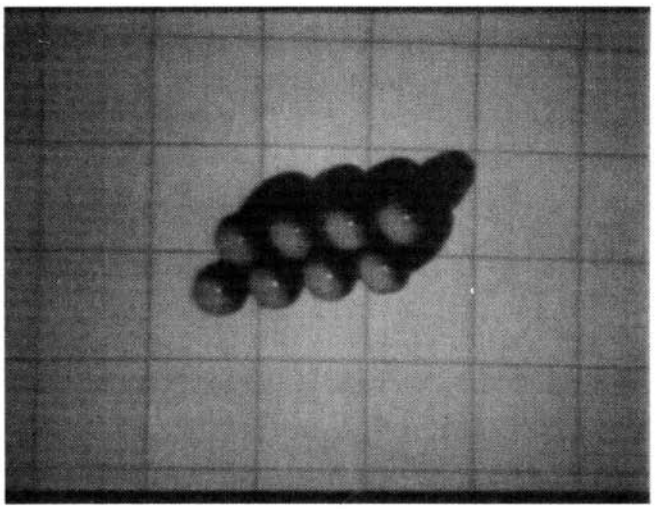
— BARR - SLIDE 21 —



— BARR - SLIDE 22 —



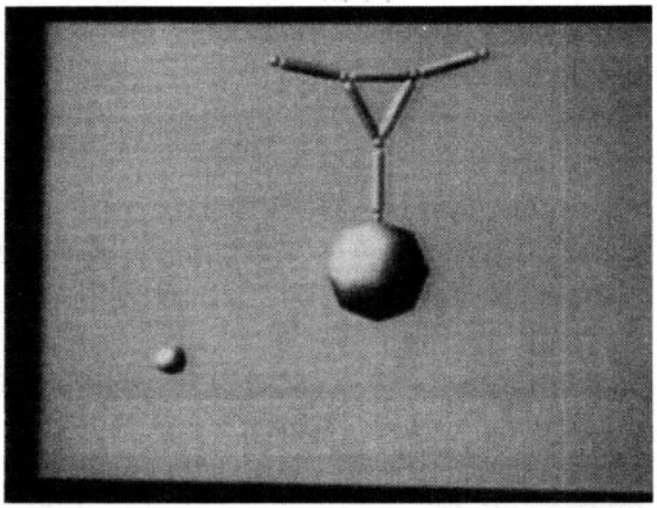
— BARR - SLIDE 23 —



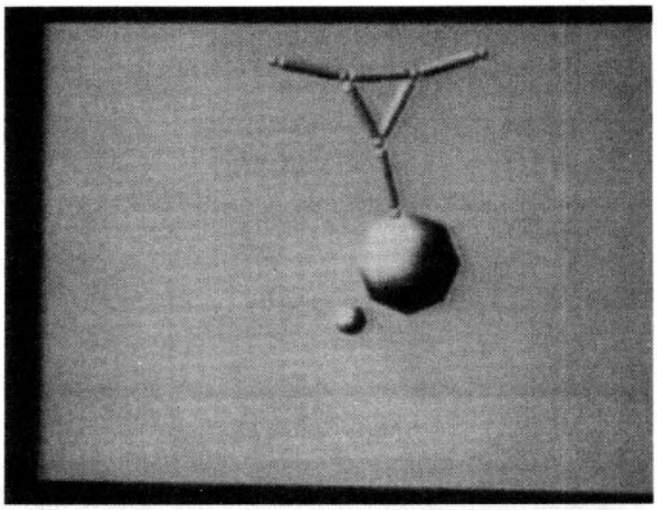
— BARR - SLIDE 24 —

The next level of abstraction is to add constraints. If I have constraints, I can connect my objects together and have them do

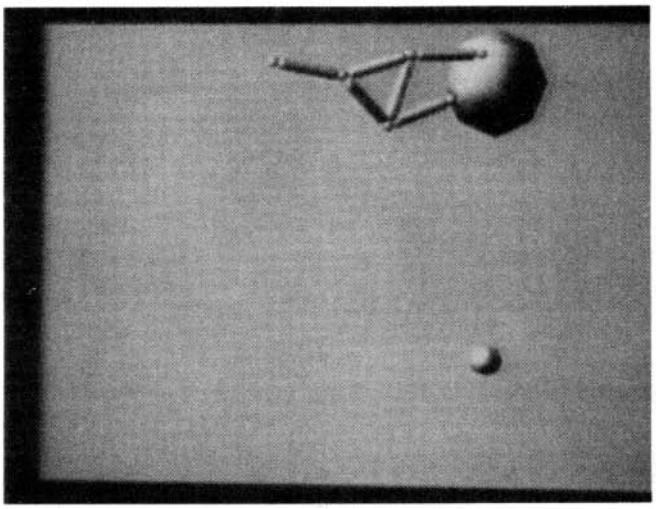
what I want -- or at least do what I say I want. In the slides, we're just saying to the balls, hook this way, hook that way, connect, and be tangent.



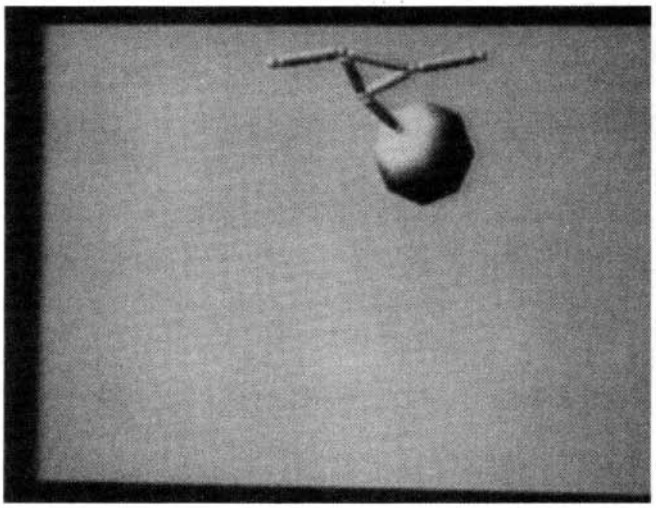
— BARR - SLIDE 25 —



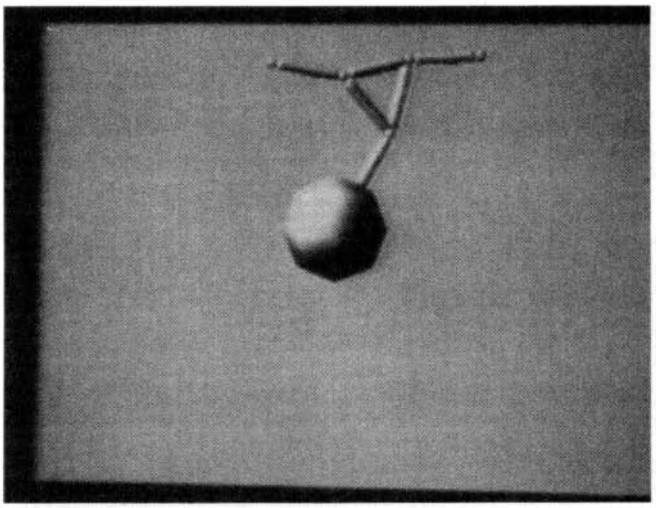
— BARR - SLIDE 26 —



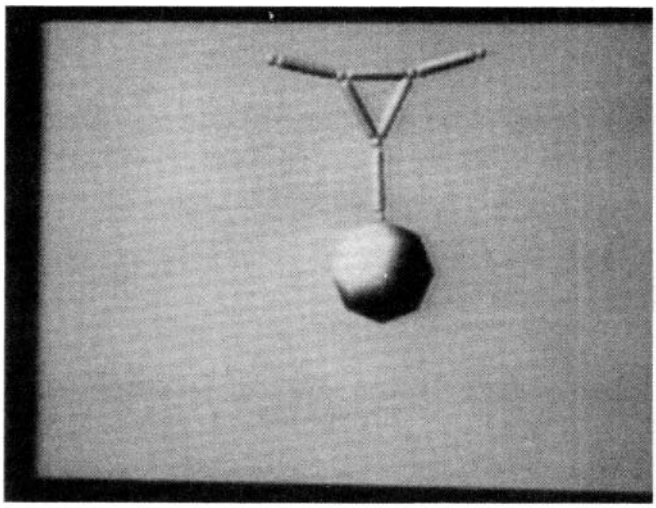
— BARR - SLIDE 27 —



— BARR - SLIDE 28 —

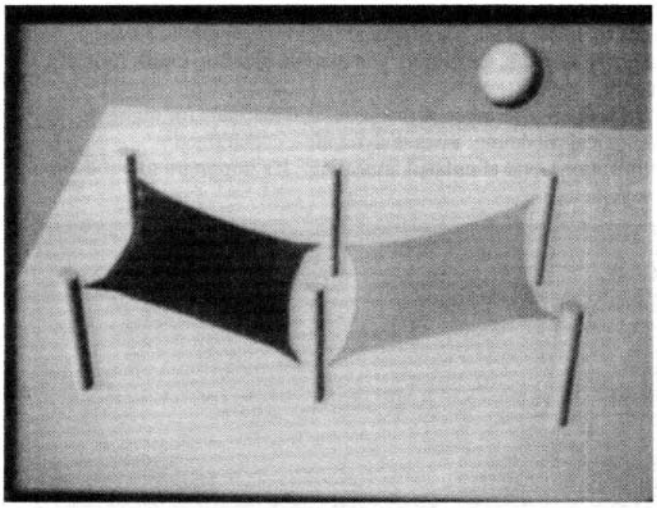


— BARR - SLIDE 29 —

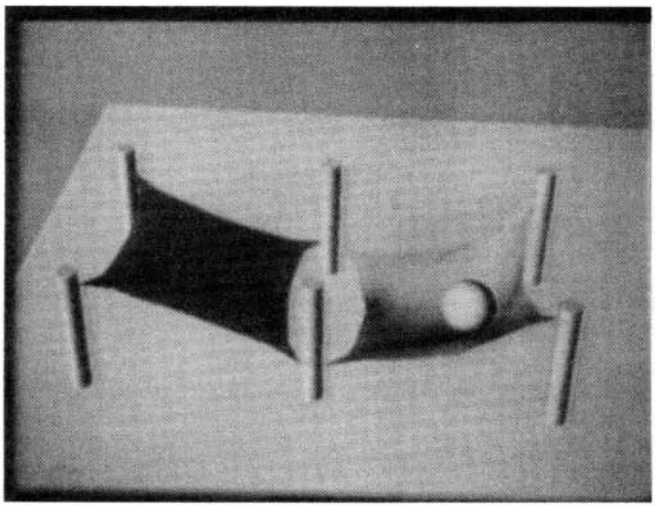


— BARR - SLIDE 30 —

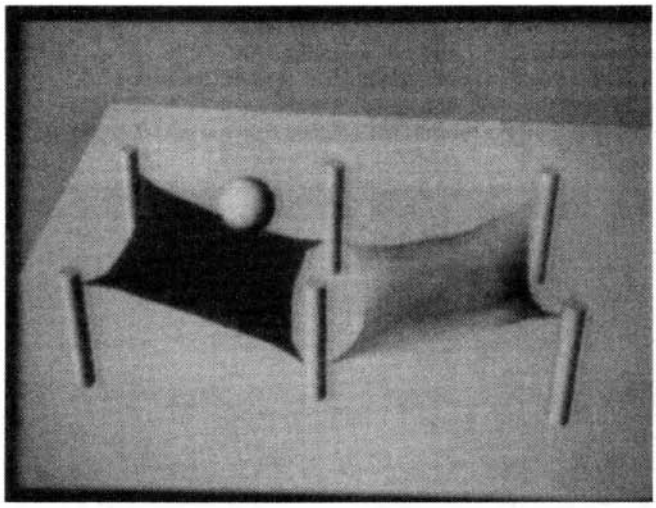
Here we're saying these balls should collide but the constraints should be met. You don't want to program in the physics by hand for doing that; you want it to happen in some automatic way.



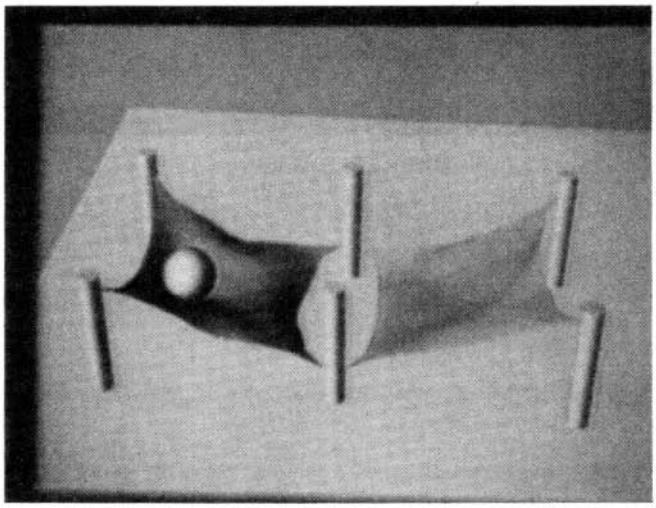
— BARR - SLIDE 31 —



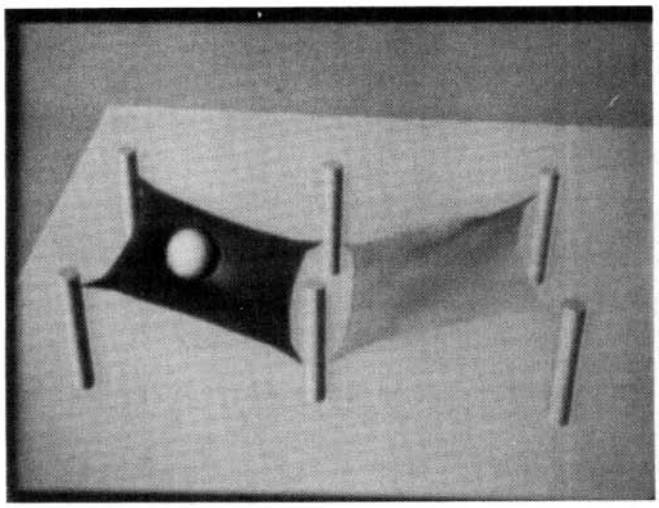
— BARR - SLIDE 32 —



— BARR - SLIDE 33 —



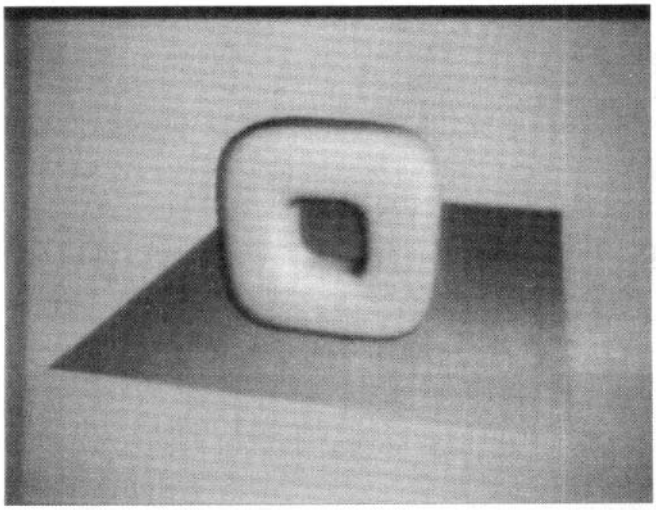
— BARR - SLIDE 34 —



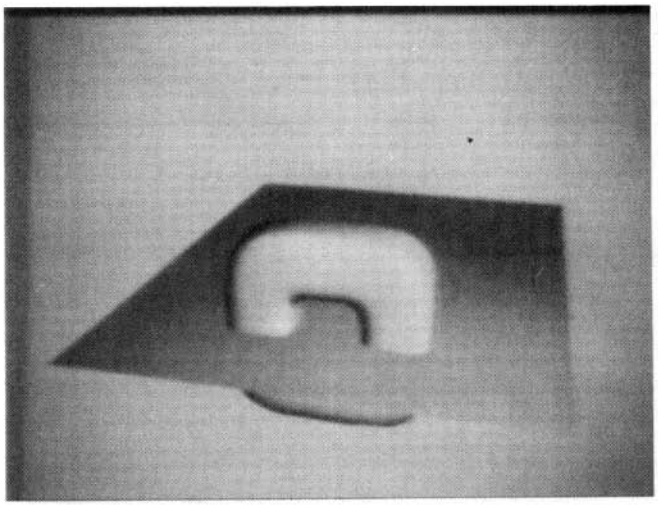
— BARR - SLIDE 35 —

Just like what you want here is the ball not to pass through the membrane.

When you don't use a teleological method, you use an indirect method. So that means that you have to fiddle with your parameters until the result is the accident that you get what you want.



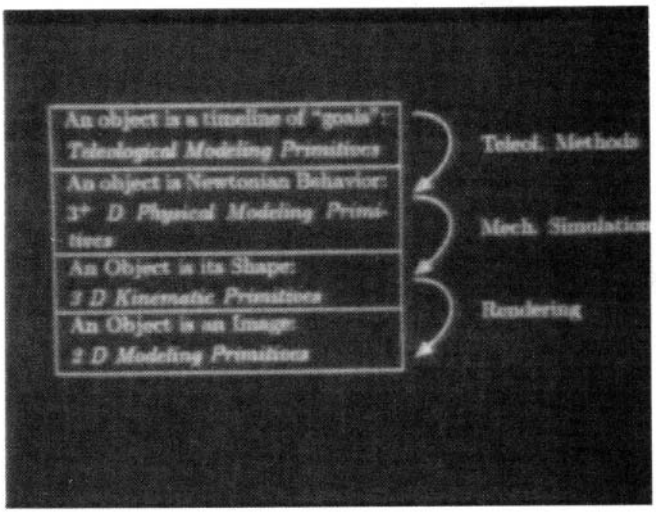
— BARR - SLIDE 36 —



— BARR - SLIDE 37 —

For instance, let's say that I indirectly want the doughnut to be on the table and I'm going to directly specify the doughnut's position. I can say, ``Put the doughnut at a particular location,'' and the computer will do it, but it might penetrate the table. Ideally, what you want is to put the doughnut on the table and that means let it fall in the gravitational field and it will dissipate its energy. Using this technique, you can fill up a bowl with fruit and whatnot.

There are a number of mathematical methods for doing teleological modeling: inverse dynamics, constrained optimizations, and simulated annealing. It's important to put them all together.



— BARR - SLIDE 38 —

A new pipeline will be developed in graphics hardware. This pipeline will consist of four parts. The users will interact with the constraints, which describe what the users want. The next level is the physically-based level. You go from goals to physics, using constraints. The next level is shape. You go from forces to shape via simulation. The last level is an image. You go from shape to shading using rendering. This is a new graphics pipeline. And the bottom two layers are where we are now. In fact, originally, graphics just had the absolute bottom layer: An object is an image. Now they have: An object is an image and shape.

It took us a little while to realize how to really use the physics layer. I remember talking to Lance Williams a few years ago. We were making an Omnimax film and I told Lance that the right way to do everything in animation is to use physics. And Lance said, ``I don't know, Al. I don't think so.'' I certainly was convinced that there was no other way.

That Omnimax movie is being presented tonight at the Science Museum, and the interesting thing about it is that I was simulating the swimming motions of creatures and I thought that I had done my job by doing the physics of the swimming. I did it correctly. I had the camera swooping through this flock of swimming things and by the time the camera got to where it was going to be, they had all swam away. Well, that doesn't seem right. So, I aimed them at the trajectory of the camera. The camera swooped through them and they swam behind the camera. So after fiddling with this for a while, I realized yes, Lance is right. There's something more than physics. There is the specification of what you want.

That's what teleological modeling is. It lets you control the physics and get what you want in a mathematically guaranteed way. Whatever you don't say that you want, you're not guaranteed to get. There might be a happy accident in which the physics might accidentally give you what you want, but it won't be guaranteed, unless you use a mathematically guaranteed method.

We're going to show a little bit of animation here. What we're first going to see is an attempt at connecting objects together using rubber band forces. The yellow arrow is a force and it drags the rod over to the nail, but the rod doesn't really get there. Now we're going to add a second rod and connect it to the first rod. Whoops, the first rod pulled off the nail! So you can see that making something out of rubber band forces looks like it's made out of real rubber bands when you turn on gravity. If you want to guarantee that the objects will be held together, you need a smarter force than this sort of rubber band force that is small when you're close to fulfilling a constraint and large when you're far from fulfilling the constraint.

So here's basically the inverse dynamics approach: the teleological approach. We say: Hook this point on the rod to that point in the middle. The green lines displayed are the velocities of the points. Notice that a radial force connects the rod to the nail.

Of course, when you remove the constraint force the rod will fly off into space. When you add a second rod and ask it to hook to the first, they now stay hooked together, unlike the previous case. There's no friction unless you ask for friction. When you suddenly ask for gravity then the object will fall and stay hooked together. The forces adapt to whatever they need to be in order to hold the objects together.

You can assemble objects. Here's a tower that's putting itself together. We're just saying hook this object to that object. Hook this strut to that strut, hook that strut to that rod.

In this next example we're going to see two towers, and we're going to connect the set of chain links between them. We ask the constraints to hook up the chain links, nose to tail. So the physics and the shape and the pixels on the screen are all byproducts. We didn't calculate those by hand. They're all byproducts of the teleological commands, which is just hook the links together, nose to tail, and hook the end points to the tips of the towers.

Here you see a more lively, more snakey chain. Although the commands to create these animations are easy to use, it took a great deal of work for our group to create the substrate. Ronen Barzel of our group and John Platt and many other people in our group worked very hard to make the underlying substrate. So even though it took three lines of code to specify this whole movie, there are thousands of lines of code at the substrate level.

In principle you can use these methods to control real physical objects. You can have spacecraft that can dock automatically, as is illustrated here.

But this is just the beginning. I think that we're seeing the very beginning of making complex systems of objects that do what we want. We've just scratched the surface. When we have hardware that can do this in real time and when we can render it and see it in real time, we will take all of these capabilities for granted and wonder how could anyone every have lived back in the old primitive days when you couldn't even have an object bounce on the screen right in front or you or drop a piece of jello on the table.

So I'm going to end my talk here with the wiggling jello. Thanks very much. Our next speaker is Professor David Zeltzer of the MIT Media Lab.

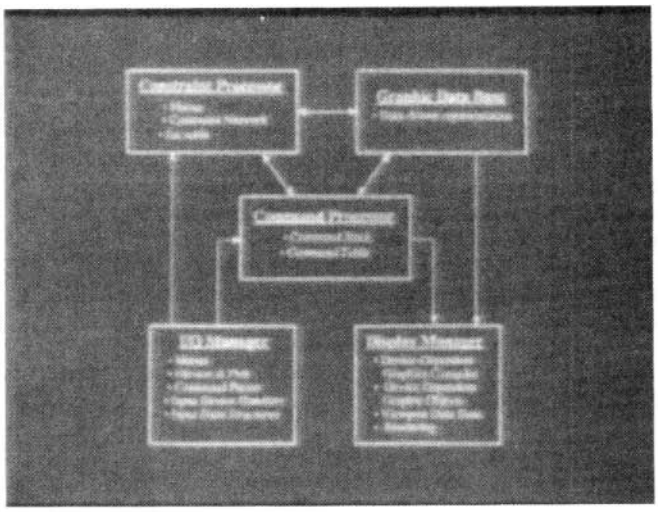
David Zeltzer
MIT Media Laboratory

Good morning. I think physically-based modeling is a crucial element of putting together convincing micro worlds. Can I have the first slide, please?

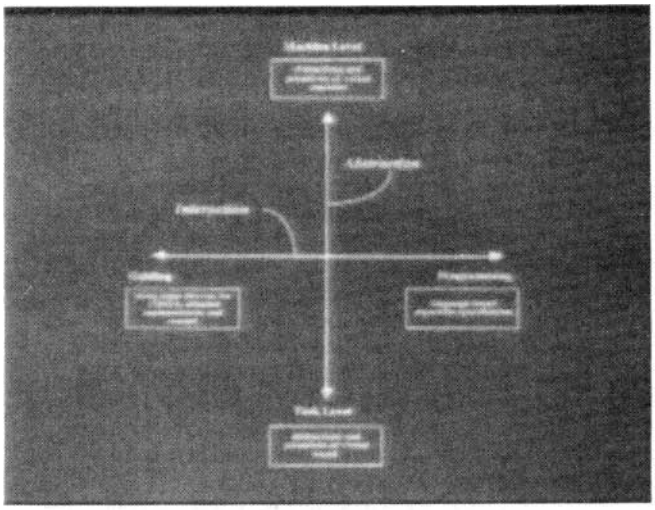
What I'm going to talk about is in some sense a continuation of the notion of abstractions for physically-based modeling and micro worlds that Al Barr was just talking about.

We're working on something we're calling an integrated graphical simulation platform. That is to say, a workstation that knows a lot about the physical world, that provides a medium for users in a variety of applications to experiment and explore a variety of computational models. We're interested more in allowing people to observe the behaviors of autonomous agents and objects rather than convincing them in some sort of synthetic reality. Here are a couple of applications. Fred Brooks has been doing some wonderful work in his lab at UNC involving virtual experiments in molecular docking. We're also interested in providing people with a medium for learning and exploring -- for looking at computational models and peeling back the levels of detail as they gain confidence in their understanding at each level of representation.

It's important to us to allow people the means not only to control computational models, but also to define them and represent them and modify them. If a scientist is studying a computational model of some process -- motor control, for example -- we'd like to allow him to program that model and insert it into the micro world to control some agent and observe its effect. So, we're interested in exploring the kinds of windows people can have on these computational models.



— ZELTZER - SLIDE 1 —



— ZELTZER - SLIDE 2 —

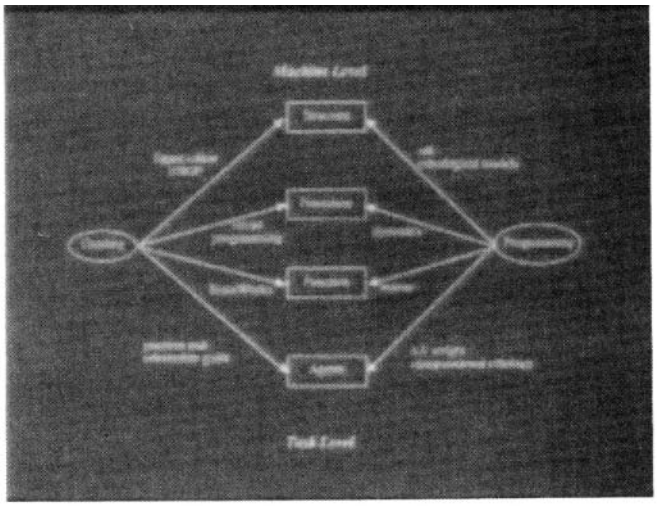
Here is a block diagram of the system that I'll tell you a bit about in a few minutes. There are a number of modules. These three here are rather standard device dependent and device independent models for graphics. They provide the substrate of the system. Then we've provided protocols for talking about constraints and for plugging in a variety of application modules.

Some of the I/O devices that we've been able to work with are of course the VPL Data Glove and the Spatial Systems SpaceBall. Recently our system's been ported to Scott Fisher's lab at NASA AMES and we've been able to plug the head mounted display into it. This fall, we're looking forward to starting to program the force feedback joystick. This is a three degree of freedom joystick with a range of motion about like this. We're doing this work in collaboration with the mechanical engineering department at MIT.

I've been thinking about abstraction mechanisms for these micro worlds. This is not a new idea, as you've seen. Al Barr's been thinking about it too. It's also quite common in Artificial Intelligence as a means for understanding how to represent and control devices for a variety of purposes -- among them, diagnostic reasoning. If there's a system that you're controlling and it breaks, you'd like the system to be able to help you figure out what's wrong with it.

In particular, I've been interested in abstractions for representing and controlling objects. There are four kinds of abstraction mechanisms that you see there. Perhaps the most important one is the functional abstraction which provides you a way of decomposing large, unconstrained and largely uncontrollable systems with very many degrees of freedom, into a number of constrained controllable subsystems with a few degrees of freedom. So we can constrain a system for a particular behavior such as walking or reaching, and then we can allow agents to achieve various kinds of motor goals by composing these functional subsystems -- either in parallel or in sequence.

I believe there are a couple of ways of thinking about the problems of controlling and representing objects. I think there are in this perspective a couple of orthogonal axes. This axis represents means of interacting and controlling objects from direct manipulation, on this end, to algorithmic specification, on this end. The other axis represents the representational abstractions. I think Al Barr's abstractions are another orthogonal axis which represents levels of representational detail. There are many dimensions in which we could abstract these objects, depending on the particular application or reason for which we're working with these objects.



— ZELTZER - SLIDE 3 —

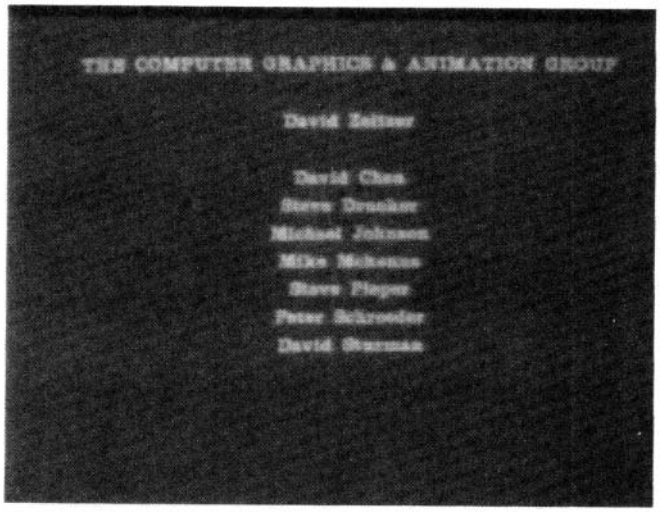
So, in particular, we can think about the different interaction modes combined with the abstraction levels, to give us a set of useful windows for interacting with our computational models. For example, we can use direct manipulation to control structures directly, and this gives us a way of describing to the system how objects are put together and their kinematic and dynamic attributes. At the same time we can write programs that operate

directly on structures, as you've seen, and this I think gives us something like "teleological modeling."

On the other hand, as we compose more useful functional behavior repertoires, we can interact with more complicated agents using the same interaction paradigm. So, if we use direct manipulation techniques to interact with agents, we have something called task level interaction where we can point to an agent and say: "I want you to go over there," and leave it for the agent to figure out how to do that.

Or, if we use programming to interact directly with agents, rather than conventional programs, we can interact with them in terms of natural language or constrained natural language scripts.

I don't have a lot of time; perhaps in the question period we can talk more about those kinds of abstractions.



— ZELTZER - SLIDE 4 —

These are the people in the group who are largely responsible for a lot of the work that you're going to see, and this is a word from our sponsors. I'd like to show you a couple of short clips right now.

The first piece you're going to see is some dynamic modeling of, again, human facial tissue. Steve Pieper, one of my grad students, is doing this, and we're working in collaboration with Dr. Joe Rosen at Stanford University and Scott Fisher at NASA AMES as part of an effort to put together a surgical simulator. So, you're seeing three layers that represent human facial tissue. There are muscles, as in Keith Waters' and Demetri's model of skin. Here you can see the muscles contract to make the skin bulge in various directions. We can also simulate various surgical procedures.

Let me fast forward. This looks better in fast forward anyway. Here's a procedure called a Z-plasty, which is a plastic surgery procedure for changing the dimensions of the surface area of a piece of tissue. That work was done using the same system I'm about to show you now. This has sound, so I'll shut up and let the tape run for a few minutes to give you an idea of the kinds of things we've been doing.

— VIDEO TAPE TRANSCRIPTION —
narrator: David Sturman

Here at the Computer Graphics Group at the MIT Media Lab, we're doing research in simulated environments. We're using forward simulation techniques in which we set up the environment and then let things go and see what happens. We've developed a testbed for this kind of animation and simulation. Generically, we call it an Integrated Graphics Simulation Platform. For local historical reasons, we have named this particular implementation Bolio.

Bolio is written entirely in C. It runs on Hewlett-Packard 9000 series workstations. In this demonstration, we're using a series 9000 835 workstation with Turbo SRX graphics hardware. Bolio consists of a core set of routines that handle device input, graphical output, and maintain the environmental data base. These routines can be accessed and shared by independent simulation modules. The modules communicate with each other and the objects in the environment through a network of constraints. Using this constraint network, simulation modules modify object attributes, such as size, position, orientation, velocity, and color, and these changes in attributes trigger other simulation modules which, in turn, modify other objects. The emergent behavior of the network generates the simulated environment.

Here we've set up 4 objects connected by spring constraints to simulate a South American bola. When I grab an object, it triggers a spring constraint which moves the other objects. That movement triggers spring constraints which move the objects again, and so it goes.

Along with the usual keyboard, mouse, tablet, we have two input devices that allow direct manipulation of the environment. The VPL Data Glove has optical fibers along the fingers to sense finger bend angles, and a Polhemus tracker that uses a magnetic field source and sensor to yield the relative orientation and position of the glove with respect to the source. A Spatial Systems SpaceBall senses forces and torques applied to it and gives us six degrees of freedom.

— END OF VIDEO TAPE TRANSCRIPTION —

I think you've seen enough to get a good idea of the work we're doing. Let me now turn the podium over to Professor Andy Witkin from Carnegie Mellon University.

Andrew Witkin
Carnegie Mellon University

Like Dave Zeltzer, I'm interested in doing real time interactive physical simulation, but I have a somewhat different angle on it. Rather than the traditional role of simulation as quantitative prediction -- what will happen if I pick this up and throw it; where will it go -- I'm interested in using real time physics as a modeling medium, analogous to what you do with modeling clay when you sculpt a shape out of it. The physical properties of the clay are a convenient way to get the shape you want. They're not part of the thing you're modeling. So I guess I'll start immediately with the tape -- if we can roll the tape -- and talk over it.

The basic idea is to be able to start with a purely geometric object -- we'll start with curves in the plane which have no physical properties. There's no sense in which a circle has physical behavior -- it's undefined. However, we want to give it physical behavior in an automatic, consistent way which we can derive just from the geometric equation that you need to draw the thing. What that lets us do is turn a purely geometric object into something we can manipulate in a direct physical way. So this circle has degrees of freedom for position and radius and we can pull on it and get any size circle we want and place it wherever we wish. Here, we get any ellipse we want. Rather than worrying about the parameters, we can just frob the thing directly. This is my personal favorite -- a spiral.

So we are able to obtain this physical behavior automatically from the geometric equation that defines the curve. A basic way we do that is by giving it a physical interpretation that says there's uniform damping and negligible mass along the length of the curve.

Now there's a kind of constraint that you can put on these objects that's trivial. It involves freezing one or more of the parameters. So here we freeze the radius of this circle. Now it's a rigid circle. And we can make a sort of inverse punching bag by attaching a spring. Now when we unfreeze the radius, we get very different behavior. These are all real time things, by the way.

To impose constraints on objects, we use a classical method of Lagrange multipliers. Here, we're illustrating that for a particle constrained to travel on a circle. The yellow arrow is the force we apply and the green arrow is a constraint force. You can see how the net force vanishes when we're trying to pull the circle directly off the circle. The resultant force -- the blue one -- is simply being projected on to the tangent to the circle. So what we're doing is calculating the force we need to add in to counterbalance the component that's trying to pull us off the circle. It's that simple. That extends to much more complicated systems and it involves solving a system of linear equations to do that projection.

Now, there's a little bit more to it, because if that's all you did you would drift off because of accumulating numerical error. So, we add feedback. Here the particle has drifted off the circle and the feedback pulls it back. So feedback gives you something that's very stable and robust and fast.

Using that constraint method coupled with the dynamics of the object, you can start to build little things. Here is the same old circle we saw before. Now, rather than attaching it by a string, we nail it in place. So there is an ether nail there, and you see the circle can go anywhere you want it to as long as that particular point stays exactly where it is. Now we can attach things together to obtain something that has some constrained degrees of freedom. You don't have to worry explicitly about what those degrees of freedom are. You just pull on it and it goes to where you want it to go. This is a nice way to build and manipulate things.

You can start to do more complicated things using the same mathematical machinery and make contraptions. Here is a circle again. We're doing the same things we did before, but we are doing it from scratch. So now we have attached things together. Now we can start to reshape things and add some more objects and make linkages and watch them go. This all running in real time on a Silicon Graphics Personal Iris, by the way.

So you can draw things with this, design things with it, do constructive geometric proofs and such. So there it goes. It behaves the way it is supposed to. As you can see, it is all damped behavior because we have assumed that these objects don't weigh much and that the drag forces dominate. So you can add more and more stuff. Here we get to use one of the spirals. It has all sorts of nasty parameters that you don't even want to think about. It would be very unpleasant to try to control an object like that directly by turning the control knobs.

These methods are a way of just worrying about how the object looks and where you want things to be, rather than trying to figure out what you have to do to the random scaling parameters to join angles and things like that to make things go where you want them to. So now we have built this thing and it moves around. It does whatever you tell it to, subject to the constraints.

One of the things you can do with this method is control for key frame animation. It is a very different thing than doing animation using physics to determine the motion. This is using physics to determine where things are going to by pulling them there and hooking them there.

You can do some more abstract things with interactive dynamics. One of those is to do optimization. If you have some function you want to minimize, then you can turn that into a force that is minus the gradient of the function you want to minimize. That gives you something that always pulls towards the nearest local minimum.

Here we have made a little scattergram and we are minimizing a locally weighted distance from the model to each dot. So each dot is exerting an attractive force. You can pull the model off and then when you let go, it gets sucked in. Here you can see that. It is a strictly interactive thing because what the user is doing is picking the model up and putting it near the desired solution and then you let go and it rolls into the energy wells. Here is the same thing with an ellipse that is going to automatically fit itself to that sort of "O" there when you turn on the force. So this is the optimal ellipse fit. Since it is hard to optimize non-linear functions globally, if things fall into the wrong local minimum, you just pick up the model and help it out by putting it near the minimum you are looking for.

You see, these things are stable attractors, if you let go, the model snaps back as long as it is not to far away. One of the applications of that idea is to interactively fit models accurately to the shape and motion of things in real live images. Here is a nice image. We have a little line that is being attracted to edges. It is the same idea except that it is attracted to points of high contrast in the image. You can see that if you let go of the line and if it is reasonably close to start with, the line will get sucked into the edge and stay there. If you perturb it a little bit, it will come back. You can track motion that way.

This illustrates snakes, an earlier work that Michael Kass, Demetri Terzopoulos, and I did at Schlumberger Palo Alto Research. Snakes are springy pieces of wire. They are a type of deformable model. Here, we are attracting them to edges, and since they have lots of degrees of freedom they can conform pretty much to any shape. Here you see snakes conforming to the shape of an edge. So you can blast the snake off the edge, and it will come back. It is basically the same behavior that you saw before.

Next we will see motion tracking. If you have some video, you can fit snake models interactively on the first frame, and then as you advance from frame to frame, all the energy well attractors move around and drag the snakes with them. Here is a movie of a person speaking, then we will see two snakes superimposed on the moving lips and tracking them. So you see that the snakes really lock onto the lips and follow them very well. This was the only thing that is not real time on this tape. We did this on a Symbolics lisp machine and, though it didn't take too long, it couldn't quite keep up. So there are all sorts of interesting things you can do with snake models.

Now all of this extends to 3-D and we have an initial system that Michael Gleischer, my student at CMU, has implemented.

My 3-D input device, by the way, is a mouse, which works very well. Here we have particles and we can connect them by distance constraints. These constraints aren't springs; they are hard constraints that are being enforced by solving a linear system. So here is a little jointed thing, that is now rigid. It is a little triangle and you can pull on it. This is again real time. So here we have made a tetrahedron and again it is rigid. Next, here is a little contraption. Those blue things are 3-D ether nails, or anchors in space. So we have attached things to them and now we have this odd little linkage that has it's degrees of freedom and we can pull it around.

Various people participated in this work, and here are their names. Now a word from our sponsor. Thank you. Our final speaker will be Dr. Jim Blinn.

James Blinn
California Institute of Technology

Well I think physically-based modeling is a terrible idea. Is that ok? They put me on this panel to cause trouble, I guess. I am not sure why they picked me to do that, but the idea is that we are supposed to have some lively discussion and dissention here. So, I am going to tell you the bad parts of physically-based modeling. Before we do that, let me show my gratuitous video tape.

Before I saw the light and realized how evil physically-based modeling is, I used to do it myself. These are some random scenes out of the Mechanical Universe. Modeling physical phenomena, especially simple ones, is fairly straightforward with the computer. A lot of the things you have seen today have been physically-based modeling of more complex phenomena.

One of the objections I have to the printed description of this panel is with the statement that physically-based modeling has been done only in the last five years. Well no, actually physically-based modeling has been done from the beginning of computer graphics. One of the first computer animations I saw was called *The Tumbling Box Movie*. It was a simulation of a box tumbling while it is in orbit around the earth. So physically-based modeling has been done more often than non-physically-based modeling, even in the early 60s.

Many things can create problems, as you can see in this simulation of an ideal gas exerting pressure on a piston. If you simulate some phenomena exactly, they just don't do what you expect. For example, we had problems with this piston in that it started oscillating up and down; because, if you only use a few atoms, you wind up with statistical irregularities interacting with the natural mode of vibration of the piston, given the spring constant of the air and the mass of the piston. And so, we just prevented the animation from going on long enough for that kind of oscillation to start building up and being obvious.

A better simulation of how atoms work is this somewhat different force field between individual atoms. Once you sort of see how that works with any two atoms, you can do it with a larger number. Here is our version of atomic jello. A single frame of this animation looks really boring, so it is kind of pointless to publish an article in a magazine about it.

Basically, with physically-based modeling, for the most part, you give the simulation some initial conditions and stand back and let it fly and see what happens. The big trick is controlling it to do what you want. There are a lot of demonstrations in the mechanical universe project of this sort of thing, for example, where we wanted to show the effect of 10 to the 23rd atoms using only 100. We had to be very careful about setting up the initial conditions so that the atoms evolved in the way that we wanted them to.

Well anyway, what sort of business does this lead to? It sort of turns animators into video game pilots. Generally, animators are used to dealing with the positions of objects. They specify the position of various key frames either by drawing explicitly or something. Physically-based modeling means that they are going to be specifying the accelerations of objects. And they have somehow to figure out what accelerations to use in order to get the position they want after the acceleration has been integrated twice.

An analogy may be made between painting and photography. Painting is the old technology of doing things manually. You have to have a lot of skill to be an artist and represent something realistically. With photography, you just aim this little box at the thing and click and a realistic picture comes out right away. You can make a similar analogy between what you call animation, or key frame animation, and simulation. Key frame animation is how it used to be done. It took a lot of skill and the animators had to know physics as well as painters had to know light and reflection and so forth, and the animators had to know physics in order to simulate it manually. Once you use computer simulation, all that is taken care of automatically for you. You no longer have to have experts to do this; now amateurs can do it too. Physically-based modeling means that now everybody can get into the act.

So there is a progression of what goes on in modeling. We've seen the progression from key frame animation, specifying positions, to physically-based modeling, which is specifying accelerations and forces and what not. The next level beyond that, as we are getting into the future, is what you would basically call psychology. You kind of give your characters motivation and tell them that they like this thing and they don't like that thing. A common phrase is "Gee we can land men on the moon but we can't learn to live together in peace and harmony." Well there is a reason for this. Landing men on the moon is really easy. That is just physics, we know how the moon operates and it is just a matter of some acceleration vectors and so forth. Living together in peace and harmony is not easy at all. We don't understand psychology well enough to be able to predict how people are going to act, and even if it is desirable, to control how they act. So as a next stage after physically-based modeling, you might consider what could be called emotionally based modeling. This is something that, for example, classical animators, like those at the Disney Studio, were real good at. They were able to put emotions into their characters.

But, if you have a computer doing this in some automatic way, it removes the animator one step further from exerting total control over the environment; animators now become like movie directors. They are dealing with something that has personality. You have to exhort your character and get your character excited about the part. You have to convince your characters to do it your way instead their own way. The characters might have temper tantrums and go off into their dressing rooms and blow lines and make mistakes and so forth.

So where do we go beyond that? Beyond that we get into meta physically-based modeling. You put your hands on the television screen and you channel the spirits of all of the past great animators and rub your crystals over the screen. When that sort of thing happens, then maybe we will all be out of business. I don't know... Thank you.

Moderator
Demetri Terzopoulos
Schlumberger Laboratory for Computer Science

I'll take this opportunity to point out that we could not possibly show all of the exciting work that's going on in physically-based modeling at this panel. I regret that the panel

could not have included several other talented researchers who have made important contributions to physically-based modeling. Having said that, I would like to open the floor microphones for audience participation. We welcome your questions, comments, flames, whatever you like. Please state your names and affiliations before asking your questions.

Q. My name is Arthur Who and I am with Mosaic Software. We make lotus compatible spreadsheets. On the metaphysical thing, there is a medium called Radio which I imagine uses the metaphysical type metaphor.

TERZOPOULOS: Is your comment directed to anyone in particular?

WHO: Well, I guess Jim Blinn talked about just imagining things and that is sort of what Radio uses.

TERZOPOULOS: Do you care to respond to that Jim?

BLINN: Sounds good to me!

TERZOPOULOS: Is there another question out there? I'm having difficulty seeing.

Q. My name is John Dunic. I am with IBM. I am directing this to either Alan Barr or Andrew Witkin. Most of the things you were showing looked like they are real time. In fact, Andy indicated that they were. But, there were small numbers of elements in your system. How many elements can you simulate before performance degrades such that you can't have real time?

BARR: In my case, it was already degraded so that it was not real time. It took about fifteen seconds a frame on a Symbolics machine, sent over the net to a Hewlett-Packard workstation and rendered frame by frame. What is interesting though is that part of the research that we have been doing over the past year or so is on this scaling problem. How do you simulate the universe in such a way that you get reasonable accuracy, yet you are not simulating the behavior of every molecule. Let's say that you want to simulate a field of grass for instance. Would you want to do elastic bodies on each blade of grass? No, you need a different abstraction to do that. My expectation is that we are going to need different kinds of physics that are just as accurate as current physics but can automatically go between the different kinds of representations.

WITKIN: For my stuff there is a more concrete answer: the things that I was doing were dominated by solving the linear system for constraints. I was using an iterative method which is essentially n-squared complexity in practice, where n is the number of elements. However, if every element in the world is connected to every other other element in the world the method turns into n-cubed. For ordinary things it is more like n-squared. But you'd like to continue adding new objects and connecting them to a few existing things. How fast is your computer? Eventually, even n-squared will be too slow, but N-squared is not really very scary. It is something you can fix by having faster computers and also with some linear systems you can probably use LU decomposition methods that are order-n, so that it would all be linear time.

DUNIC: Could you imagine connecting this up to a CAD system, for instance, and expect it to work?

WITKIN: Sure, absolutely! To do large scale things, we'll need to wait a little while. At least, I will need to wait a little while for faster machines than I currently have. The things that I am now doing in real time took a few seconds a frame for me a couple years ago with the machines I had then. So you know, things improve. It is real time technology.

TERZOPOULOS: Perhaps I can add something: With regards to CAD/CAM, deformable models appear promising as a type of computational modeling clay. We will soon be able to simulate 3-D modeling clay in real time on our graphics supercomputer class machines. In the past, the speed limitations of our machines restricted our interactive simulation to 2-D where it was only mildly interesting. Who's next?

Q. I am Dave Breem from RPI. I was wondering how physically-based modeling, as you describe it, is different from what the physicist and mathematicians and the mechanical engineers have been doing for the past 100 years, besides the fact that you are just making pictures from your models?

BLINN: The difference is that we are doing it now instead of them.

BARR: That's actually not completely correct, they are still doing it. In addition, it turns out, let us consider the physics of a particular body. How should we represent the body? For physicists it would be quite satisfactory to say, in principle, that we have elastic van der Waals forces between the different molecules. We have the covalent bonds between the molecules. You can do it all at the molecular level. Or, you can be a mechanical engineer and you could talk about the fluctuation and bending strengths and what not. See, a scientist typically cares about their discipline only and not the modeling techniques that another discipline might use.

And so there is in the future something that I will call generic scientific modeling in which you are quite happy to model the molecular behavior. Or if you need to you will model this other behavior. The difference is that were interested in the generic modeling. In terms of all of these constraints, the physicists typically are happy with the description --- let us call it the U=0 equation -- the unworldliness = zero equation. It is not necessarily a description that can be easily implemented.

This example that I gave requesting a sort of flexible body with non- interpenetration constraints, it takes the physicist a good long time to write down the equations of motion of that. If I were to change the abstraction it would take them a long time to react to that. I have been talking with Ronen Barzel -- we have been thinking about this. How come it is so easy to state a little piece of the model, yet it is so hard to do the actual simulations, to actually write the code. When you think about it, as Ronen and I decided, two hundred years ago, three hundred years ago, even taking a square root was difficult. So that the physics that has been done over the past three hundred years is physics as designed for use without computers. So the physics that we are designing is one that is good to use with computers. I would say that that is the difference in the physics. There is actually a different physics behind it -- a different collection of equations. So although, the actual Newtonian appearance of it is of course the same because it has to be if you are going to be presenting the real thing, the underlying equations are completely different, at least some of them.

WITKIN: There are important differences in what we are using this stuff for. I agree with Al that to be able to add in some new kind of object into your simulation and connect it to other objects without having to go back and rewrite all your code is, maybe, good system design, but it is a comparatively new development. Also, there are things we want to do. Making movies for movies sake, for example, is not something that a physicist or mechanical engineer is going to do. The stuff I was talking about using physical methods to develop modeling media or, as Dave Zeltzer was talking about, to develop interactive micro worlds where you can play ping-pong or something. These are just different things and very often it is the same physics underneath and ultimately at least similar in the numerical methods. But what you use it for colors a lot of what you do and a lot of the technical problems that you have to solve to make things really work.

ZELTZER: I think another important difference is our emphasis on interaction in real time. As our computing tools are getting powerful enough to let physicists and mathematicians deal with the formerly intractable models, it's turning out that the ability of a scientist to apply his specialized knowledge about where the solution might lie is critical in finding solutions. Fred Brooks has a wonderful example in which he shows that using an interactive force display as well as visual cues allows scientists to find solutions in molecular docking problems interactively, while a SUN-4 for example cranked overnight was not even close to the solution. So, interaction is something that we are bringing into the problem as well.

BARR: My prediction is that there is going to be a great body of knowledge that is going to go from people on this panel and other researchers in graphics back into the physics community. I think there is a lot of good information that we will be giving them. That is my predication.

PLATT: Also, just in terms of the math, it hasn't actually been hundreds of years. Again, because of the emphasis on the computer, some of the constraint math has been around in mechanical engineering only from 1972 or so, and we have been developing it further.

BARR: Let us just consider something called solving simultaneous equations. You would think solving simultaneous equations is easy. But, when you actually try to do it on a computer it turns out that your systems become unstable. Your solutions get sent out to infinity. So, you need to use a completely different kind of solver that was only invented a few years ago, called singular value decomposition. When you don't use it, what happens is that all of your answers get turned into mush. There is a great deal of difference. You learn a lot more when you "really do it," rather just saying "U=0 "-- I wrote the equation - something like that ought to work.

TERZOPOULOS: Go ahead, Sir.

Q. I am Salim Abi-Ezzi from RPI. I direct my question to the whole panel; who ever cares can answer me. In the past we were successful in expressing the problem of displaying shape very concisely, and we came up with what we call the graphics pipeline -- Transformation, clipping, rendering. Having worked on these problems in physically-based modeling, do you think that we will be able to express the physics and constraints that are needed in a concise and generic fashion, so as be able to have hardware accelerators, for example?

BARR: You should read the PhD thesis of Devendra Kalra, hopefully coming out in the next year. Our expectation is that there will be a significant amount of progress on the problem you are addressing.

ABI-EZZI: The answer is yes?

BARR: Well, in one year, it is not finished yet. Devendra will also be talking later on this; I guess on Friday. So, if you wanted to, you might be able to speak with him personally after the talk.

TERZOPOULOS: Perhaps Andy would explain how he goes from analytic expressions, as a concise way of expressing the physics and constraints, to executable code automatically.

WITKIN: Yes, sure, that is concise for the things that I am doing. There is the geometric part of objects that we know and love -- exactly the stuff you need to draw objects. So, if it is a curve, the geometric part might be the parametric equation for the curve. The same thing for a surface. Using symbolic math, you can add a physical interpretation which says how objects are going to move. It is sort of a template that you fill out mathematically which will let you take some symbolic derivatives, make some symbolic simplifications, and then turn it into C code that goes to the compiler. These templates involve mathematically extremely concise descriptions that can be converted automatically into stuff that you can execute.

Also, as far as accelerating the things we do, a lot of the low-level operations that go on are main stream. When you are solving linear systems there are a lot of dot products, matrix multiplies -- exactly the things that people who are programming supercomputers are usually worrying about, so in some cases there may be neat ways to set things up and make them go fast. They may be quite generic and not special to what we are doing.

TERZOPOULOS: Ok, go ahead please.

Q. My name is Terry Boult. I am from Columbia University. My question is directed at the entire panel, but particularly to those who are interested in trying to actually model the physics, especially for animation of the body, like in the facial animation that Demetri showed. Is your goal to actually have animators start specifying force profiles for all the muscles that control a person's face or a person's arm? If not, why are you going through physical modeling as a means of giving someone just another type of clay to work with. Why not start simplifying long before you have to start solving finite element equations or partial differential equations?

TERZOPOULOS: Well, our goal in facial and body animation is to develop process models that control individual muscles. The animator will interact with the model at the high level of abstraction. He will give a high level command, let's say, "smile, broadly." The muscle process will coordinate individual muscle contractions to initiate the expression, the physical layer will propagate forces through facial tissue, the tissue deformation will modify geometry, the geometry will be rendered, and the animator will see a happy face.

Why are we going through physical modeling? In large part because you automatically get more realism that way, and often it's critical. Keith Waters developed a face model two or three years ago which was a purely geometric surface warped by muscles under kinematic control. It is fast and looks fairly good, and for certain applications it may be sufficient. For example, if you are trying do band limited teleconferencing, so at one end you take pictures, a movie, of the face of a speaker, you analyze the pictures in real time to extract a few parameters for a face model, you transmit the parameters over a low bandwidth channel, and then, using the extracted parameters, you reconstruct and animate the face at each receiver so that others may "see" the speaker, it may be sufficient to do that using a purely geometric face model. However, if you are making a feature involving animated characters, such as Marilyn Monroe and Humphrey Bogart in the Universite de Montreal production *Rendezvous a Montreal*, and you want a close up of faces, geometric face models suffer from too many artifacts. People can be very critical of human faces. I think that to make a really good human face you have to model some of the anatomy and some of the underlying physics.

WITKIN: I have a one word answer to that question. It is: control. If you look at what really happens when people and animals move around, do tasks, and so on, you will see an interaction between their own physical selves and the physical environment, and what happens in their brains to control this interaction. Of course, if you were going to make a physical model of someone walking or talking or anything like that, to try and do that at the level of actually specifying the forces that the muscles are applying would be a disaster. It would be hopeless. The point is that you can solve for the forces that need to be applied to accomplish a task. That is an interaction between the job that is being done and the mechanical situation in which it is being done.

Can I show the video tape? I just happen to have one to illustrate what I mean. It was a take off on Luxo Junior, that maybe some of you have seen. We define a jumping critter and give it muscles that it can control. We tell it to go from here to there. Then we indicate the optimal way to do that, how it can use its mechanical resources, its muscles, to do the job. From this specification, you get really nice structured motion that has both physical realism and goal-orientedness, by specifying something that in the end winds up looking a lot like key-framing. You are saying, be here now and be here then.

ZELTZER: Let me give another answer while Andy is setting up the video tape. That is, that animation in the conventional sense is only one thing you might want to do with these systems. In the piece I showed of the facial tissue simulation, the purpose is to provide surgeons a means for planning surgical techniques. So, of course, faithful physical modeling is critical, otherwise the application is entirely worthless. It is not just the case that these techniques are only devoted to generating animations that tell stories.

BARR: I think that what Jim Blinn was saying is actually quite exciting. This emotionally based modeling is really quite real. There is a brain biologist, John Allman, at Cal Tech who is quite interested in how emotions can control the movements of the faces. Certainly, if you want to express some sort of emotion with your medium, it would be hopeless to specify it with forces.

ZELTZER: In fact, physiologists have developed a system called the facial action control system in which they have categorized the muscles of the face. It is pretty well known which muscles are involved in creating which expressions.

BARR: They can even tell which is a real smile and which is not

ZELTZER: That's right. So this is a tool providing economical control of facial expressions.

TERZOPOULOS: Andy has a video tape to show.

WITKIN: Ok, let's take a video break! This is work that Mike Kass and I did at Schlumberger Palo Alto Research. If you look at the way Luxo Junior jumped, this is a obvious take off on that. There is a lot of structure in there. All we are saying here is that Luxo should start at the beginning and stop at the end. We have a full mechanical model of Luxo, and we say: do it with minimal muscle power. Then we have an iterative solution that goes from a stupid initial version of the motion that does not look real, to something that does look real. We are showing the solution process with a sequence of strobed images. So we are going from the initial version to the final solution.

Here we are going to play the solution back, and we get a jump. Look at all the stuff that goes on in there. There is squash and stretch, and all of that, which comes out as part of the physical solution. We give it basically two key frames to do all that. There it is in slow motion. Then since it is a physical thing, you can change the motion in sensible, intelligible ways by changing the physical situation a little bit. We changed the mass of the base and it is all exaggerated. And look at that in slow motion. You can take that as far as you want to. Here's a hurdle jump with one more constraint that says clear the hurdle. In the slow motion, notice how Luxo gets the extra height -- by scrunching, rather than by jumping higher, which is the sensible and energy efficient way to do it. Mike Kass programmed a ski jump.

So this was pretty hard to do; the mathematics is a little bit rough. We're solving a variational optimization. But eventually I think we'll be able to package this into something that, when we're done, starts to look kind of like a key frame system again -- even though what goes on inside is a lot of mathematics.

TERZOPOULOS: We have time for one or two more questions.

Q. I have a question for Jim Blinn. I'm Ronen Barzel from Cal Tech, and you sort of said physically-based modeling is a crummy idea. I figured I'd pick up that gauntlet. You made a really nice analogy between painting and photography. I really do like the analogy; I think it's really valid. But would you extend the analogy and say that cameras are a really crummy idea?

BLINN: When they're aimed at me they are, yes! There is an effect of this that you see, in that before cameras were invented, painters primarily painted realistic scenes and they were hired to paint portraits of people and so forth. When cameras came about, cameras took over that process. Instead of having a painter, you had a photographer. And so it was no longer commercially viable for painters to do realistic paintings, and it was no longer necessary. It sort of freed the painters to go off and paint weird abstract things and they no longer had to focus on reality -- "photographic reality." They were able to start exploring things, because anybody with some training can copy reality, while somebody with maybe more imagination was needed to do something interesting abstractly. So maybe the fact that physically-based modeling comes along and takes over some of the mechanical operations that animators have been doing manually might free the animators to do more interesting abstract things.

TERZOPOULOS: One more quick question, please.

Q. John Williams, MIT. I think physically-based modeling seems like a really great area, but I feel there's a kind of conspiracy of silence about the actual physics and modeling, the mechanics. As you're probably well aware, there've been techniques around from the early '70s, like the finite element method, the boundary integral method, finite differences. I don't really see anything different being proposed now -- if the aim is to do physical simulation. If you want to really predict how the physics is going to move through time. It seems to me that the real benefit here is on throwing away the physics and saying we're willing to do inaccurate physics. We're willing to make some approximations which the mechanical engineers and civil engineers wouldn't make. And it seems to me, then, we can get this interactive behavior, which in fact makes the models really useful. Perhaps the panel can comment on this silence about finite elements.

BARR: I gave a physically-based tutorial last year that included John Abell who spoke about finite elements. Finite elements are integrally involved in what we're doing. It's one of the mathematical methods that we have at our disposal, even for solving certain integral equations for synthesizing the swimming motions of objects. I would say it's not fair to characterize the bulk of what we're doing as "inaccurate modeling" -- that would not allow us to make predictions. We're building on that previous work. So, if there's a conspiracy of silence, it's because we're making reference to this work in our publications and perhaps people are not picking up on it. But singular value decomposition is a technique, Gear's method for stiff equations. These are some of the tools that we're using.

WILLIAMS: But if you look at all the examples that are given, they're all very deformable-type models and there's a good reason for that, because if you have very stiff materials, they're much more difficult to analyze. I do it myself. I mean, I like floppy models because I can get the answer out in no time at all. Whereas a piece of metal, it's tough, and the animation in this year's Computer Graphics Theater of the falling teapot which breaks (*Tipsy Turvy*). That's very deformable and there's a good reason. If you try to do it with a very stiff, brittle material, it will take you hours on a Cray.

BARR: There's a talk this year at SIGGRAPH called Modal Analysis by Sandy Pentland.
Q. I'm the co-author on that.
BARR: Now that's good stuff.
TERZOPOULOS: I'm afraid our time is up, so I'm forced to terminate the discussion. My apologies to those of you who didn't get a chance to ask your questions. I would like to thank the panelists and to thank you for coming to the panel..