CSE557 Graphics Final Project Detailed Description

Cartoon Physics

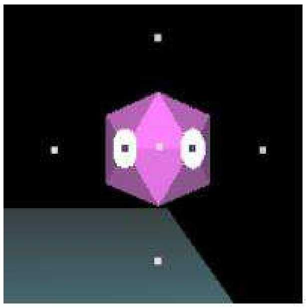
Jiwon Kim
Karen Liu

1. Model

   The model we build is an icosahedron (which is a Platonic solid composed of twenty faces that span twelve vertices, each face of which is an equilateral triangle). Between each pair of vertices, we connect them with a spring. We also let all vertices connect to the center of icosahedron with twelve springs. In the real world, each spring has its maximum and minimum lengths that constrain the motion of two ending points. For example, the one ending point cant go through the other one and keep moving. However, the simulation of particle system doesn't embed those constraints. The model could reach an unwanted balanced state and stay there if we didn't explicitly specify some constraints. We have two approaches to maintain the correct shape of our model. One is adding some extra particles and springs and the other is implementing angular spring constraint.

   (a) Extra Particles

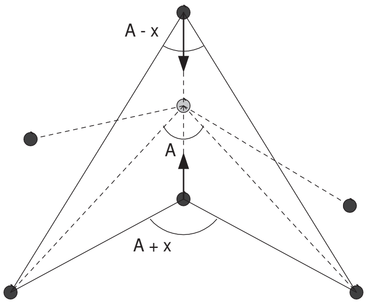
   The idea of adding extra particles and springs is very straightforward. First, we add another particle which doesnt connect to any of other vertices as a reference particle. This reference particle only moves according to the external force such as gravity or the force animators specify. We also add six vertices around the model like this:

   Each particle at one side connects to the four closest vertices on the model. Every time we render the model, those six vertices correct their positions to a fixed offset from the center. When the model moves suddenly, it is those particles that keep the model goes back to its original shape. After couples of experiments, we give the springs connected to those six particles a constant three times more elastic than those on the model.

   (b) Angular Constraints

   We also tried adding extra constraints for the angles formed by each pair of springs. We defined a force whose magnitude is proportional to the angular displacement and has an opposite direction (see figure below).

As can be expected, it shows a more stable behavior when returning to the original shape after deformation. However, it constrained the motion of each particle more than necessary, sometimes resulting in almost rigid body-like motions (as can be seen in the third animation sequence of the artifact).

2. Inertia and Gravity

   Inertia is the essence in a lot of cartoon motions. We let users to specify an starting point and an ending point, and we generate all the motions in between. The main force that directs the motion is along the vector of the starting point and ending point. The inertia is generated by the initial velocity which is opposite to the force we apply on the model. Users can also specify different mass to different parts of the model. We let the bottom part have the heaviest weight to make it more cartoon-like. When the model reaches the middle point between starting point and ending point, we make the force have the opposite direction so that the speed of the model starts decreasing.

   Every time we refresh the model, we check whether its base is on the ground as opposed to staying in the air. If its base is not on the ground, we apply the gravity on it. This act can be postponed for a while depending on how the story goes (e.g. the cartoon character doesn't fall until he looks down). The delay is an adjustable parameter to users.

3. Collision

   We implemented two different situations - collision into a static object like wall, and collision between two mobile objects. The first case in turn was implemented in two different ways - with pure physics (the 4th sequence in artifact) and cartoon physics (the 5th sequence). The second type of collision has not been fully implemented; it only works for the particular example we use (the last sequence).

   We used the model with angular constraints for realistic collision. As mentioned before, however, these constraints are too strong for deformation to occur. So the resulting animation almost looks as if it is a completely rigid object. We also tried using the less stable model without angular constraints, but this time its constraints were too weak - the object turns into a very grotesque shape after bouncing off the wall, and never comes back to the original shape.

   For cartoon-like collision, since we wanted the object to go completely flat on the wall, we simply changed the velocity of each particle to zero. We didn't elaborate further to make it spread wider to imitate volume preservation, so it is basically equivalent to projecting it perpendicularly onto the wall. In the second collision, however, the length of flattened 2-D projection does depend on the speed of the colliding object.

   After a certain amount of time, it goes back to its original position and shape. This was also done in purely manual fashion; no physics is involved here.