

# Project Proposals

## Intro

We are investigating balloon simulation using a mass-spring system. We want to try and simulate the elasticity latex/rubbery material of balloons and how they float in the air. We will look into how balloons keep their intended shape when inflated with spring constraints and see how inflating the balloon more will be simulated with those constraints. We will also look into anchoring the balloon with a string and how it affects the buoyancy simulation.

## Related Works

In “Semi-Realistic Balloon Simulation”, Tarantino attempts to simulate balloons with a mass-spring system. Tarantino does not use various spring types (structural, shear, flexion, etc), but does vary the spring constants. However, Tarantino does model a viscosity force. The balloon model does not include any Provot correction, but does include a correction where if a spring is overextended, the simulation stops applying new forces to the spring. We can investigate using Tarantino’s correction method, a method similar provot correction, or no correction at all. Tarantino also allows for balloons to burst, but we will not do this in our implementation, since we do not want to kill our balloons.

Not quite similar to ours, doesn’t appear to include different spring types (structural, shear, etc), but does include viscosity forces. No provot correction, but disables forces if springs are overstretched. We can investigate if we should use provot correction, the correction described in the paper, or not at all. The paper also mentions varying spring constants throughout the balloon allows for the balloon to blow up more in certain spots. We can try this approach, or also see if using angular springs will help achieve an effect similar to this.

One idea of making our mass spring system is drawing from “Fast Simulation of Mass-Spring Systems”. In this paper, they reapproach how to simulate cloth and mass spring systems with a simpler model and less calculations to make the system faster. The biggest difference with the algorithm is how spring forces are calculated, instead of Provot correction and collecting sums of forces, it’s just done with an optimized reduction of Hooke’s law, which reduces the amount of calculations done per particle greatly (Liu et al., 2013). However, they mention that their implementation does not really take into account all 3 kinds of springs in traditional cloth simulation, which results in a less-faithful simulation of cloth. But since we are not simulating cloth, this less-faithful simulation conveniently makes things look more rubbery, which is what we want because we are simulating balloons. We also might need to simplify the springs with this method simply to help lessen the load as we make buoyancy calculations too.

With the advantage of simpler spring calculations, we can fit in buoyancy calculations. The approach that Jinwook Kim and his colleagues proposed in their paper. Their algorithm is

actually a bit more complicated than we need because our first goal is to have a traditionally shaped balloon to work first, but their approach is quite clever. It takes advantage of the rendered geometry and uses a “slice” of it to and approximates how mass is distributed within it to calculate how it should bounce in water (Kim et al., 2006). So, since we have predetermined shapes for our model, we can easily put that data in our data structure to use to make bobbing look more realistic. This might make things easier for possible interaction with other objects and simulating the resulting movement of other objects.

### 🎈 Written Example 🎈

We would like to be able to load a quad mesh and inflate it and have it float in it's scene. We will not simulate the inflation of the ballon, and our first goal is to have it floating in an empty white space. Our most trivial example will be a single inflated sphere bobbing in nothingness and being able to inflate more according to the spring constraints. It has nothing to collide to and there is no air or wind physics to deal with. A more complex example would be creating an inflated balloon from a squirrel model that Michael has and have it's starting position be off the equilibrium, which will result in it bobbing back into equilibrium, as well as being able to inflate more into something more oddly shaped than a squirrel. To make our lives easier, we will automatically consider the “string” of the balloon it's center of mass. So imagine moving the string on a balloon and watching the balloon move back into place.

### 🎈 Division of Labor and Timeline 🎈

We have 2 major things to do:

1. Be able to load arbitrary quad mesh object files and generate springs between the vertices/particles.(estimated time: ~1 week)
2. Applying forces to the vertices/particles. This includes spring forces to simulate the elasticity of the balloon and the buoyant forces resulting from the string of the balloon. We will use Explicit Euler integration, and explore more complicated integration methods if time permits. (estimated time: ~1 week)

We have a stretch goal of collision detection and resolution. If we have time to think about that we will implement it.

Michael will do the quad mesh loading, and Annie will do the force application. To make things smoother, Annie will also work with Michael on the particle data structure so force implementation is easier.

## Bibliography

1. Tarantino, Paul. "Semi-Realistic Balloon Simulation." [alumni.soe.ucsc.edu/~pault/262paper/262paper.pdf](http://alumni.soe.ucsc.edu/~pault/262paper/262paper.pdf).
2. Liu, Tiantian, et al. "Fast Simulation of Mass-Spring Systems." *ACM Transactions on Graphics*, vol. 32, no. 6, 2013, pp. 1–7., doi:10.1145/2508363.2508406.
3. Kim, Jinwook, et al. "Fast GPU Computation of the Mass Properties of a General Shape and Its Application to Buoyancy Simulation." *The Visual Computer*, vol. 22, no. 9-11, 2006, pp. 856–864., doi:10.1007/s00371-006-0071-x.