CS207 Final Project: The Parachute Simulator

Doug Keating

Motivation:

Computational methods are uniquely suited to simulating problems that cannot be solved analytically. One such problem is the motion of a falling object affected by drag. The force acting on a falling body due to air resistance can be modeled as:

$$F_D = \frac{1}{2}\rho v^2 C_D A$$

$$Where:\ \rho = The\ density\ of\ air, C_D = The\ drag\ coefficient\ of\ the\ falling\ object, A = The\ cross\ sectional\ area\ of the\ object, and\ v = The\ object's\ velocity.$$

Therefore, the total force acting on the object and determining its downards acceleration is:

$$F = \frac{1}{2}\rho v^2 C_D A - gm,$$

$$Where\ g = Gravitational\ force, and\ m = The\ mass\ of\ the\ falling\ object.$$

Because the object's acceleration at time t, its velocity is not solvable analytically. Furthermore, a realistic parachute employed to slow a falling object will not have a constant surface area orthognal to the objects motion. Therefore, ***A*** will also vary widely as the surface of the parachute moves and deforms due to the effect of drag. My goal for the parachute simulator was to use a meshed mass-spring model to realistically simulate these dynamics for several different parachute shapes and object masses.

Integration & Extensions:

The Parachute Simulator is enabled by the integration of functionality from several extensions :

- Johan and Brendan's JB_Parallel extension to speed up the position and velocity calculations done during each Euler step, and smoothly render the 3,586 nodes and 10,240 edges that make up the three test parachutes.
- Serguei and George's Mesh-Spring model to simulate the surfaces of each parachute and allow for calculation of the effect of drag.
- Yoonji and Muhammad's SDLViewer Extension to provide listener events and enable key commands that affect the simulation and viewer.

Each extension provided helpful information on how to implement the corresponding features, and each worked immediately when example programs were compiled individually. However, integrating all of the functionalities for use in the simulation was fairly challenging due to variations in datastructures, method implementation, and type definitions. While my Mesh implementation worked well with JB_Parallel and the SDLViewer listeners, it did not interface well with the meshed mass-spring extension, and the meshed mass-spring extension could not easily interface with JB_Parallel. After trying a variety of combinations and representations, the easiest means of interfacing all three sets of code was by adapting Serguei and George's Mesh representation that underlies the meshed mass-spring model and supplementing additional iterator functionality to better work with JB_Parallel. The entire process was

made more difficult by the complicated compiler errors that replaced those provided by Clang after switching compilers to implement parallel functionality. However, it provided a useful exercise in adapting and interfacing code by differing authors.

Program Design:

The parachute simulator is designed to calculate the effects of drag on the motion of an object suspended from a parachute by realistically modeling the effect of drag on a changing surface area. For the simulation, an accurate density for air (~1.2kg/m3) and drag coefficient of a falling plane (Cd = 1.28) were used to provide the highest level of realism. The force is then scaled by the orthographic projection of the mesh triangle's area onto its velocity vector, allowing accurate representation of the effect of drag even on cells that are not orthogonal to the direction of movement.

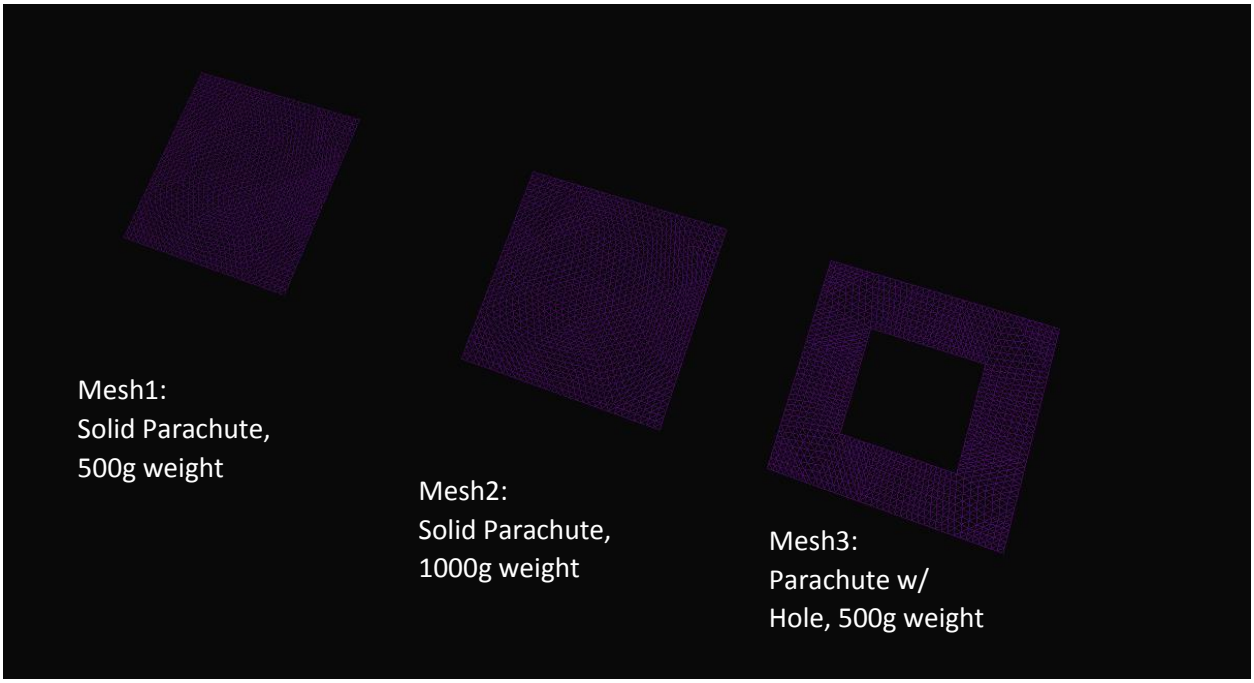
Three parachute models are tested in the simulator: a square parachute suspending a 500g mass, a square parachute suspending a 500g mass, and a square parachute with a central hole suspending a 500mg mass, and each parachute occupies a flat area of 1m2.

Instructions for Running the Simulation:

Build and start the program using the following terminal prompts:

```
make parachute_sim
./parachute_sim data/chutes.nodes  data/chutes.tris
```

The viewer will load in the three test meshes shown below:



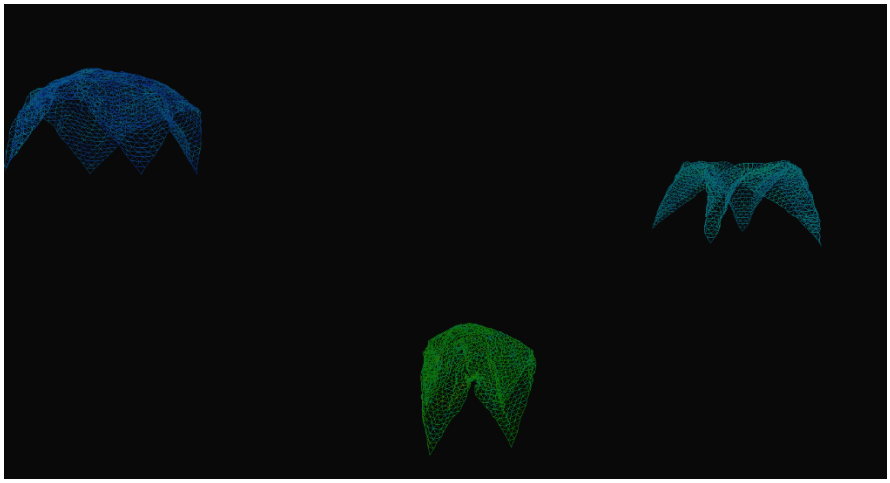
The simulation will initially be in a paused state. Press "p" to un-pause, and begin the simulation.

Note: There are several additional keyboard commands that can be used to modify the simulation:

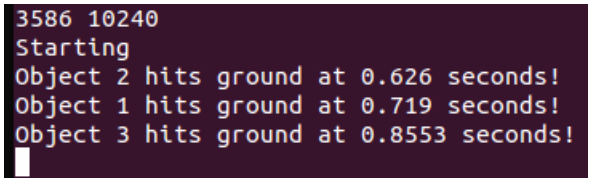
Key	Function
"p"	Pause and un-pause the simulation
"1"	Increase the mass hanging from Parachute 1 by 100g
"2"	Increase the mass hanging from Parachute 2 by 100g
"3"	Increase the mass hanging from Parachute 3 by 100g
"4"	Decrease the mass hanging from Parachute 1 by 100g (Will not allow negative mass)
"5"	Decrease the mass hanging from Parachute 2 by 100g (Will not allow negative mass)
"6"	Decrease the mass hanging from Parachute 3 by 100g (Will not allow negative mass)

Warning! Be careful not to press 1, 2, or 3, as the impulse caused by suddenly adding substantial weight can cause the meshes to become unstable.

As the simulation progresses, each parachute will move based on the calculated effects of forces due to air resistance and gravity, and take on a unique shape. The relative velocity at each point of the parachutes can be determined by the coloring of the nodes – cool colors represent slow velocity while warmer colors represent more rapid velocities.



As the falling items reach the ground, the horizontal plane constraint will print a message noting the time of impact for each object.



These times can be used to compare the fall times of each type of parachute, as well as to compare the results of the model to numerical solutions for falling objects facing drag.

Directions for Further Development:

Non-Rectangular Meshes:

Rectangular meshes do not provide an ideal representation of a parachute, as they do not efficiently occupy the maximum surface area once deformed by the effects of drag. A non-rectangular mesh such as a circle or a hexagon would provide a more optimized representation, and provide a simulation that is truer to real life parachute design.

Improved Air Movement Modeling:

The Parachute simulator does not account for air that is captured under the parachutes canopy, and prevents the canopies from filling with air to provide the maximum possible surface area. The accuracy of the simulation would benefit from a method of calculating the effects of trapped air, and pressure on surfaces that is not dependent on their velocity or direction. While the implementation of an Air Pressure Force centered under the canopy may be feasible, it would not account for the effect of this built up pressure on the drag forces generated by the parachute, and a more complex model may be required.

Rigid Body Object Payload:

The current implementation of the simulator models the weight hanging from each parachute as gravitational forces acting on each of its four corners which are held in fixed positions. This method was chosen because attempts to model the object as a rigid-body, with appropriate mass and connected to the parachute canopy by edges with a high stiffness resulted in the destruction of the meshes due to the effect of impulse on the linking edges. Successfully overcoming this problem and creating a suspended model of the object would increase realism and allow the calculation of forces due to tension in the 'ropes' that suspend the object that are not restricted to the z-direction.

Interesting Observation:

The most interesting phenomenon observed during the development of the parachute simulator was that of canopy collapse. In real parachute drops, opening a chute at low velocity can result in the collapse of the canopy because the reliance of drag and air resistance on velocity. Several of the test configurations that were considered displayed this behavior, and oscillated between lower and higher velocities as the parachute alternately filled and collapsed.