Cloth Simulation using Mass-Spring Model:

The goal of this project was to implement cloth simulation using the mass-spring model, where fabric is mimicked by many particles connected by many more springs. The project uses Verlet Integration to update a particle's position. It keeps track of the current position and the previous position to update future positions. The equation of updating positions is derived from position being the second derivative of acceleration. Resulting in the equation: $x = 2x_n - x_{n-1} + a_n \Delta t^2$. The equation is similar to the one in class, just a different way to implement it (no velocities).

The particle changes positions based on the forces applied on it using Newton's 2nd law: ∑F = ma. The main force for a mass-spring system is Hooke's law or the spring force, where F=-kx, the force is equal to the displacement of the spring times a stiffness constant. For cloth, there are 3 types of internal spring forces at play: structural, shearing, and bending forces. Structural force is the resistance to compression and stretching, they are the horizontal and vertical springs in our model. The shearing force is the resistance to shear, they are the diagonal springs in our model. The bending force is resistance of out-of-plane motion, we mimic this by attaching springs to points further than nearest neighbors.

Other forces can show how "cloth-like" our model is. We used external forces like gravity and wind, and internal damping force that mimics damp springs. The damping force is dependent on velocity, so in our Verlet integration, we took the $x_n - x_{n-1}$ part of our equation and damped it.

Another way to test the "cloth-like" nature of a model is with cloth collision. Due to the nature of fabric, the cloth wraps around an object in a collision so our simulation needs to mimic that. In our simulation, we simulated two types of collision, cloth collision against a sphere and cloth collision against a cube. There are multiple ways to detect collision and correct, the most realistic is to apply a normal force based on the collision, but this is hard to implement. The collision used here simply checks if a particle is going to be inside an object, if it does it will "repel" it to the edge. This creates a slippery surface (no friction).

Implementation:

The program I implemented is based on WebGL like the previous projects. The base code is taken from the Minecraft project. I implemented 3 new main classes for the mass spring system. The Particle class that keeps track of the position and the applied forces on a given particle. It also keeps track if a particle is anchored (position can not be changed). The Spring class keeps track of two particles, the rest length (initial length between the particles) and stiffness. The final MassSpring class updates the entire mass spring system and draws the cloth itself. The update function in MassSpring is the main function that is being run. It has two parts, the internal spring forces and the external forces. The spring forces loops through the springs and applies a spring force to each of the particles. The external forces loops through the particles and further applies the external forces to the particles. After the forces are added, we update the position of the particles.

The MassSpring is initialized in App.ts, where the particles are generated in a 2d grid and the springs are also generated. Three types of springs are added, horizontal and vertical for structural springs, diagonal for shear springs, and springs connecting particles two apart for bending.

The cube used for collision is the same cube used for Minecraft. The cube collision checks if the updated particle is inside the cube, if it is, return it to the previous position. To draw the sphere, we break the sphere into chunks using the longitudes and latitudes. We convert the longitudes and latitudes to the cartesian coordinates and then draw triangles with these positions. The sphere collision implemented is the slippery surface. If a particle is inside a sphere, it is repelled to the edge of the sphere.

I also implemented a Settings HUD that allows the user to change the parameters of the cloth simulation without changing the code directly. This HUD is created using basic HTML and Javascript.