Physics - Cloth Simulation - Zheng Yu Bu - Post Mortem

For this project, due to the scope required, it was to both me and Varun's best interests to do it as a group project, both to save time and to ensure the best possible result.
My assigned tasks were to integrate Bullet Physics into my DragonDrop game framework, and to make sure the key cloth physics and cloth rendering are implemented, once that was done Varun would then be able to work on the other features such as the resizing, wind, sphere, capsules and so on.

Bullet Physics Integration

This was the first objective for me, and was a fairly important one as it would determine how the rest of the project would be handled. This was because there was another way that we could have done it, and that was to make our own 3D physics engine from scratch. After weighing our options, doing a sort of SWOT analysis of the situation, I came to the conclusion that it would not be in our best interest to do that. So bullet physics it was! That meant that I needed to get the latest version of the Bullet api from github, and try to get it to link to my OpenGL engine.
First problem I ran into with this was the linker errors, these were from trying to use the bullet .lib files and using the visual studio linker with them. There didn't seem to be a way that I could get it to work well in a clean manner, so in the end I skipped the linker part and dragged the entire bullet source files into the project. This worked fine, so I left it at that.
Once all the linker issues were resolved, the next task was to get my 3D objects to have physics. This was relatively simple as the structure of bullet was quite similar to that of box2D. This just required plugging in the positions and rotations of the physics bodies and setting a collider for all of my objects that need it. After I got a basic cube dropping onto a plane, the integration was marked as complete.

Cloth Physics Implementation

Now the problem was how I would go about doing the actual cloth physics simulation. There was two ways that were possible when using bullet, first was to use bullet's in-built softbody cloth system, and the second was to use rigidbody points that connected to each other using spring constraints in a grid-like structure to simulate the verlet integration method.
My first choice was to try the in-built softbody cloth to see how viable it would be for this project. For as much as I tried however, it did not end up working as well as I'd liked, especially since I needed an entirely different way to get it to draw than how I was used to. This and also the fact that I did not know how possible it was to apply tearing and mouse-picking to the inbuilt softbody, meant that choosing to use the built-in system seemed like a risk, and one that did not have too many benefits other than an easier implementation. So doing it using rigidbody points and springs was what I chose to do in the end.
This was done by making sphere colliders as "points" that are joined together using spring constraints. I made a Cloth class that would keep track of and manage the points, and a ClothPart class that was what each point was as a physics object.

Initially, the structure of the clothparts and springs constraints was in a grid structure with only horizontally and vertically aligned springs. This somewhat worked, but it still didn't feel like cloth, and it would fly off in weird directions after a while. I solved this by using diagonal spring constraints as well as the horizontal and vertical springs, and further tweaking the damping, limits, and frequency of the springs. After that the cloth acted a lot more like cloth. The tearing was fairly simple to do after that, by simply disabling the spring constraint if it gets dragged too apart too fast.

Cloth Mesh Implementation

Once that was done, Varun was able to start on implementing his features. The drawing of the cloth mesh became something that needed work, so I was tasked with implementing it. This mainly involved playing around with OpenGL vertex arrays and making some shaders. Each clothpart would have references to other neighbouring clothparts and uses their positions to draw a quad that has a texture on it. The tearing makes the mesh point return to the origin position making the mesh in that area appear as if its been torn off.