Since my honors thesis proposal, I have slightly changed the direction of my research. I have previously mentioned that CharacterMixer will be used to create novel characters by swapping parts and interpolating existing characters. Now, I have narrowed the research goal to ”interpolate between two 3D characters with different mesh and skeleton topologies while preserving an underlying rig such that each blended character can be animated.” Interpolation is an important shape editing operation, but prior methods have limitations when applied to rigged characters: they either ignore the rig (making interpolated characters no longer posable) or use a fixed rig and mesh topology. To handle different mesh topologies, CharacterMixer uses a signed distance field (SDF) representation of character shapes, with one SDF per rig bone. To handle different skeleton topologies, it computes a hierarchical correspondence between source and target character skeletons and interpolates the SDFs of corresponding bones. This correspondence also allows the creation of a single “augmented skeleton” for posing intermediate blended characters. I have produced qualitative results and shown that my method, CharacterMixer, outperforms state-of-the-art optimal-transport-based interpolation method, while preserving a rig for animation purposes during interpolation. In summary, my contributions are:

- An algorithm for computing hierarchical correspondence between two character skeletons
- An interpolation method for blending between two characters’ geometries
- A technique for posing interpolated characters using a single skeleton derived from both the source and destination rigs

Figure 1 shows the key idea of CharacterMixer. In the top row, I show an example of interpolating Squirtle to Blastoise with a fixed pose throughout interpolation. The bottom row shows the same interpolation, but where the character’s pose changes continuously throughout the interpolation. The two characters have different surface mesh topologies and different rig skeletons (e.g. Blastoise has shoulder-mounted water cannons and a one-jointed tail; Squirtle has no cannons and a two-jointed tail).
In terms of the system itself, Figure 2 illustrates CharacterMixer’s pipeline. Given a pair of rigged source and destination characters in rest pose, CharacterMixer produces posable interpolated characters. To produce an intermediate character, CharacterMixer takes three steps. First, it finds hierarchical correspondences between the bones of the two characters using recursively defined cost functions. Second, it creates an augmented skeleton using the corresponding pairs. The augmented skeleton serves a proxy that guides geometry interpolation between topologically-different characters and supports user interaction. Third, CharacterMixer generates geometry for each bone in the augmented skeleton by interpolating between corresponding character part SDFs. Users may select an intermediate blended character and pose its augmented skeleton. Upon user input, CharacterMixer retargets the pose to the source and destination skeletons by propagating bone transformations. It then computes locally deformed SDFs for both source and destination characters and again uses the augmented skeleton to guide geometry interpolation.

I have compared CharacterMixer with a state-of-the-art optimal-transport-based interpolation method that uses convolutional wasserstein distances, which is referred to as “ConvWasser” in the figures below. Fig. 3 compares how well CharacterMixer interpolates rest-pose characters compared to a state-of-the-art, rig-oblivious method for shape interpolation based on optimal transport. CharacterMixer produces higher-quality interpolations. Fig. 4 shows several interpolation sequences produced by CharacterMixer. All of these results involve interpolating between characters whose rigs require 1-to-many correspondences.

This semester, I would like to make the skeleton correspondence algorithm more robust, as it sometimes fails to consider the semantics of bones during the matching process. What’s more, CharacterMixer theoretically not only support discretely sampled SDFs but also neural SDFs. My work has been based on the former presentation. I have experimented with interpolating neural SDFs, but the results are far worse than discretely sampled SDFs. I would like to keep exploring neural SDFs for CharacterMixer as neural representations would greatly help decrease shape generation time.
Figure 3: Comparison with ConvWasser. In the top two rows, the two characters being interpolated have skeletons in 1-to-1 correspondence. In the bottom two rows, corresponding the two characters' skeletons requires 1-to-many correspondences.
Figure 4: For each character pair, the top row shows shape interpolation with a fixed rest pose, and the row below shows shape interpolation with varying poses.