We are selecting Louisa's codebase. We came to the consensus that Louisa's codebase was further along, but Maggie's codebase was cleaner and less redundant. The final decision came down to which drawback would be easier to resolve. We think it will be a better use of our time to focus on refactoring than to implement new functionality, because this will allow us to have interesting conversations on design, rather than having to focus our discussion on game play specific to Tsuro.

In the end, it was decided that it was in our best interest to work on code that is further along. Louisa's former team had completed all the assignments, while Maggie's team had not run a tournament with machine players.

Many of the functions required to run a tournament connect multiple pieces across the game (i.e. they must keep track of multiple turns and ensure the board state stays consistent). While Maggie's code functions well for playing one turn, some current design choices may not map well to a connected system. For example, Maggie's implementation of the board keeps track of current player positions on the board. However it does not keep track of Splayers, so it does not know which positions are active vs. eliminated. Splayers, on the other hand, do not have knowledge of their position, and the two are linked with a Splayer ID. While this implementation worked when playTurn was called once or twice in the test cases, there is the potential that it might not maintain proper game state across a tournament. There is the potential that eliminated players might continue moving, or that other unexpected behavior might occur.

For these reasons and more, working to bring Maggie's codebase up to speed might expose higher level flaws in implementation that Maggie's former team hadn't looked into. These might put our new team behind as we work to move forward with our game during the remainder of the quarter.

We can still take advantage of Maggie's cleaner codebase by using it as a template to refactor Louisa's code. This is possible because the high-level structure of our implementations is very similar. As a result, we have confidence that it will be easier to transfer clean code from Maggie's codebase to Louisa's, than to build new functionality from scratch to augment Maggie's.