Discussion

Changes between MileStone A and B

When doing milestone B, I added a MapToAngle class to generate a map from bean in yaml to angle. Instances of this class are used in feature initialization, which supports code reuse.

I added several fields in CityFeature and FarmFeature, including vertex positions and all neighbor edges, etc. These added fields could fix the merging problem of cities and scoring problems of farms. In all the feature classes, I added methods which supports angle rotate and position set when the associated tile is successfully placed. MergeTwoFeatures is used to merge separate features, and updateStatus will check if the feature is complete, if complete, then the dominant player's score will increased immediately.

Also, I wrote an interface and remove the abstract class for all features to better return all the new features list when player place a tile on board, also the different methods in different features will delegate methods in interface.

In the player class, I added placeFollower method to substract 1 unusedfollower of this player.

I removed follower object, and used hashmap to store the follower information in feature class, which is essentially the relationship of player and number of follower the player has placed on the feature.

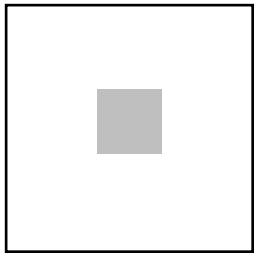
In game and board classes, I added several important methods such as getNewFeatures, updateStatus, and updateFinalScore. When validating the placement of the follower, list of new features associated with the newly placed tile will be iterated and checked. Also updateStatus() will be called after each round, and updateFinalScore() will be called to calculate scores of incomplete features.

Changes between MileStone B and C

When writing gui, I didn't make much change to core implementation. To facilitate calling the method in game class from gui, I removed some unnecessary parameters in several methods such as player input in placeTile method.

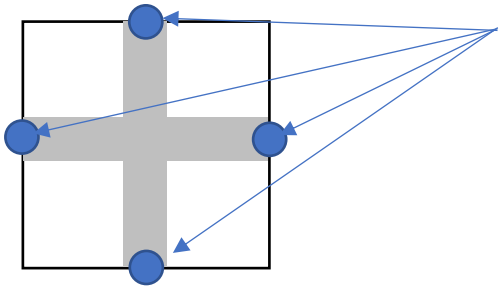
The most important part in gui design is how to place follower on a specific feature. In order for the player to directly click on the feature, I designed the following steps to check the location of the follower:

1. Search if there is a cloister on the newly placed tile.
   If there is a cloister, then check if the click point is in the shade area of the button.
   If it's in the shade area, the follower is successfully placed in the cloister feature.



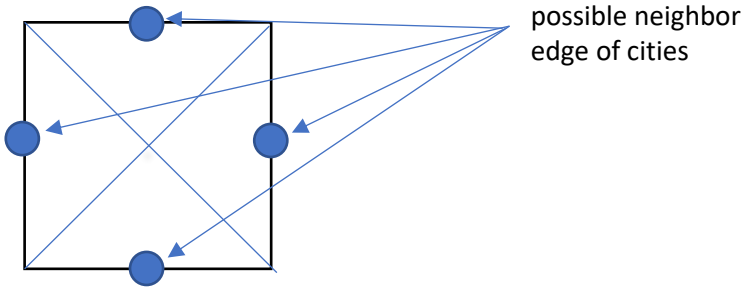
cloister area

2. Search if there are road features associated with the newly placed tile.
   If there are road features, then check if the click point is in the shade area of the button.
   If it's in the shade area, then find the closest neighbor edge position of each road feature.
   If there exist one closest neighbor edge, which distance to the click point is less than ½ of the tile length, then the follower is successfully placed in the road feature.



road area

3. Search if there are city features associated with the newly placed tile.
   If there are city features, find the closest neighbor edge position of each city feature.
   If there exist one closest neighbor edge, which distance to the click point is less than ½ of the tile length, then the follower is successfully placed in the road feature.



4. Search if there are farm features associated with the newly placed tile.
   If there are farm features, find the closest vertex position of each farm feature.
   If there exist one closest vertex position, which distance to the click point is less than 1/√2 of the tile length, then the follower is successfully placed in the farm feature.