Sprint 3 Reflections:

This sprint presented a new frontier of challenges to our team. To implement each component of this sprint successfully, our paths often had to diverge to complete conceptually difficult implementations.

To keep track of our progress throughout the sprint, our group used Trello. This application allows for the creation of task cards, and these cards can be linked to pull requests made in GitHub. The cards move through progressive states which indicate how much work has been completed on each card. Thanks to Trello, our implementations were planned out much better compared with Sprint 2.

Throughout our implementation of Sprint 3, our group faced a multitude of challenges. The first and most reoccurring challenge dealt with collision detection and responses with objects drawn in the game. While the logic to detect collision between two objects is straightforward, the shear amount of checking that went into detection for all drawn objects made the process more difficult. Our solution to this problem came in the form of creating global lists differentiated by object type. Each list which required collision detection would have coordinates of their elements compared, and coordinates that overlapped indicated collisions to trigger a response. While this solution works well for now, we will likely refactor the lists so that they are tied to the rooms in which sprites are spawned in. This logic ties into our next issue that we faced with level loading.

While collisions were a challenge for our group, we also faced the task of creating many different rooms and parsing the items inside of each room. To accomplish this task, our group decided to use XMLTrees for object specifications of objects to be drawn, and these items would be parsed in an XML parser so that object information could be added to corresponding global lists. This task was a tedious process because it required lots of repetitive code and verification of coordinate passing to ensure that all sprites and texture appeared where they were expected. However, after getting the parser to work successfully, the refactoring of object spawn locations was straightforward.

In future sprints, our group plans on reducing the accessibility of our fields and properties so that they are only available where necessary. Our code metrics indicate that we have maintainable code, but we are aware that there is much room for improvement.