Problem Statement – Currently there exists no single remedy that caters to providing players with information about their paint level, air level, and their teammates’ locations. As such, players are forced to divide their attention amongst the action in front of them, and the awkward actions of yelling up and down the skirmish lines to get a rough picture of where immediate team members are located, and refocusing attention to the actual equipment in order to obtain information on paint and air levels. The P.E.T.E.R.S. product aims to provide players with a solitary tool that grants players access to all of this information by utilizing an array of sensors to gather equipment and positional data, displaying that data to the wearer/user, and also communicating that data to other users in the field of play.

Background Research (Requirements) – In order to obtain a system that satisfies this problem statement, these requirements were derived. First, the system shall collect real-time data on a user’s current air tank pressure and the paintball level inside the user’s hopper. In addition, the system shall also be able to track a user’s location within the playing field. This raw provisional and location data shall be processed and packaged by the system and subsequently transmitted to surrounding players and displayed to the user via the HUD.

Fall Progress (Software) – A large amount of focus this past quarter has been placed on the moving map overlay as this will be the most difficult to implement. The moving map will be essentially what it sounds like: a map that continually updates its presentation based on a player’s location relative to his/her surroundings and other players so as to create a big picture of the playing field. In terms of overall GUI design as well as the entire software architecture, the emphasis has been on modularity and allowing for as much flexibility as possible with regards to decisions that may be made down the line which may impact how the software will be utilized. For instance, the decision is still ongoing as to whether the air and paint level information will warrant their own GUI subpanels or if that information will simply be overlaid on top of the main, moving map GUI. To this end, anything (classes, functions, or otherwise) that is capable of being separated into smaller, functional components has been modulated in line with that notion. This will enable a sort of “plug-and-play” ability in terms of adding or subtracting code components in the main classes, will result in those main classes being significantly shorter, and will ease the burden of debugging. At this point, most abstract functionality has been developed in the form of the base GUI constructor, the map overlay constructor which constructs a map from stored tiles, a camera class that defines the current view characteristics, the paint and air level overlay constructors supporting multiple layouts, and an icon class. Arguably the most important component of the GUI framework is the utility class to convert longitude/latitude information into pixel position information. There is no doubt that all classes will see a certain degree of change throughout the life of the project, but I anticipate the majority of it to be within this utility class.