**Spring Tactics Abstract**

Kobe Kamin

My partner and I began our capstone with a turn-based single player strategy game we’d built prior. The game was simple, lacking basics such as animation or sound. My primary goal for this capstone project was to learn new skills related to networking, and practice game development techniques such as rapid prototyping and code reengineering. This was accomplished through reengineering our game from a single player to multiplayer experience in addition to making it a more complete product. This was done using Unity, Unity Netcode for Game Objects, Unity Relay, and Itch.io as a publisher.

The most difficult issue I faced was that of player connectivity and, more significantly, synchronizing game states for players across a network. Another big problem was getting game balance, readability, and usability up to a high standard. As developers who both knew the game’s design in and out, my partner and I were unequipped to properly address these issues from a consumer standpoint.

The first problem addressed, player connectivity, was accomplished by leveraging Unity’s Netcode for Game Objects and Unity Relay. Netcode for Game Objects along with Unity Transport allowed us to send information to connected players. This solution alone was insufficient though, as we could only connect port forwarded players. This need was circumvented through Unity Relay which, as a server, acts as an intermediary between connected players, negating the need for direct connection. The issue of game state synchronization was handled through remote procedure calls. With ServerRPCs, we could execute integral game code consistently and securely then relay data back to users via Client RPCs. The final problem, entirely consumer oriented, was addressed through frequent and regular playtests with three other capstone teams.

Our project met all goals, resulting in a fully ported, playable, and standalone multiplayer game published on Itch.io. Porting a single-player game to multiplayer involved extensive refactoring, addressing the learning goal Process Reengineering Techniques. The goal of Network Design was heavily necessary during this porting process and beyond. Finally, Rapid Prototyping was practiced throughout the quarter by preparing for weekly meetings and iterating based on playtester feedback.

Our main difficulties were all growing pains from learning new skills and thus there are no major pitfalls I’d try to avoid if I were to do this capstone over. Given more time I think we’d focus mainly on improvements suggested by our fantastic playtesters and squashing a few persistent bugs.