**Architecture Discussion**

Our Design Pattern Decision was to use M.V.C. for our game.

We chose this pattern because we felt it effectively managed the control structure and organization as it applies to game programming, in particular this game.

Models

Our Model was the board class and all its elements (tiles, clearings, paths, monsters, chits, players, etc.). This was where we stored our game pieces and their appropriate data.

Controller

The Controller objects were split into two groups, the game object, and combatManager were responsible for controlling the game flow and decisions that the player did, while the server (and commThreads) was responsible for communicating the different actions between the players and managing the game/combatManager appropriately.

View

The view was our client (gameWindow, dialog boxes, etc.), in was responsible with displaying all the information from our board and model objects as well as any notifications the server sent.

This pattern was effective at managing object responsibilities and ownership.

Basic Control flow functioned as follows:

* View(client objects) displays information from the model(board et al)
* Controller(server/comm objects) gets input from the users and passes it off to either game or combatManager
* Controller(game/combatManager) updates the game state (Model(board et al))
* Controller(server/comm object) arranged the new game state (sending updated models to the clients)
* View displays the new model.