Project Analysis – Santorini Game

Part One – Components

To allow players around the world to play Santorini together on our servers, we will provide a client that they may download and run locally to interface with the game software. Users will implement their own AI players or GUI views of the game, which will interface with our client via a TCP connection, enabling language-agnostic development of user programs. The client will also connect to our servers separately over TCP. It will check for well-formed requests, and generate and track internal information, like player ID and game ID.

On the server side, there will be separate components to handle registration of players in a single game or a tournament, management of tournaments as a whole, and tracking the progress of individual games. The first component with which users interact is a Registration component. It will listen for connections and process the setup command(s) the user sends. It will then send the socket off to either the Tournament Manager or the Single-Game Manager.

The Single-Game Manager acts as a waiting room for users who wish to play a one-off game against a specific or random opponent. It will hold onto sockets, then pass them off in pairs when instantiating individual Games.

The Tournament Manager similarly acts as a waiting room for users who are joining Tournaments. It also tracks ongoing Tournaments, recording wins and losses and scheduling the required Games so all players in a tournament play each other. It reports the state of the Tournament to the participants at varying intervals.

The Tournament itself is a separate component that sets up the matches in a tournament, possibly according to a variable matchmaking schema. It keeps track of the results of each Game in the tournament and records players' overall standings. It generates sets of Games based on those standings, determines a final tournament winner, and notifies all participants of the final results.

The final component is the individual Game. Part of the Game functionality handles the communication of move requests and updates to the game state with two participants via their respective sockets. A separate part with additional sub-modules manages game state, components, and logic, which includes validating moves and determining when a player wins. Upon completion of the game, the Game passes the results and the socket back up to the proper manager.

Part Two – Implementation Plan

In order to build a working demo of our Santorini software, we will first develop the minimum necessary components to play a single game. This means we will build the client component, the Game component, and a very simple TCP server that listens for two connections from clients and creates a Game. The client component will support the minimum actions necessary to move and place pieces on the board, and will report back the game state. The Game code will need to enforce the Santorini rules as outlined, process moves, and report the state to a player's client after their opponent has made a move. With these basic pieces, plus a simple command-line interface or AI player component, we can demo a working system.

From there, the next step is to add the Single-Game Manager component that buffers incoming connections and creates as many games as necessary to let all interested users play. The Registration component will still simply listen for incoming connections and pass them off to the Manager.

After we demonstrate our software's ability to handle simultaneous games, we will implement the Tournament Manager that structures multiple games as a Tournament. In this step, we will allow for the creation of a single Tournament that is played to completion before another may be started. In this iteration, the Registration component (and client) will be extended to allow the user to register for a Tournament. Then, the Tournament Manager will handle all tracking of game results and scheduling of future games, until all players have played each other. It will also report tournament results and leaderboards to the users.

The final piece is the addition of multiple, concurrent tournaments. With this piece in place, it should be possible for any number of players to participate in tournaments simultaneously. We will extend the Tournament Manager to track all current tournaments. The Tournaments themselves should still be able to independently schedule their own Games and report results, and the Tournament Manager will handle all necessary meta-analysis of player performance.