**Game Board Engine**

The Game Board Engine runs as an event driven system. The engine is made of three main components: the Game Activity, the Game State Manager, and the Game Board UI Drawing Engine. When the local user makes a move the game state is updated and then the game state notifies the drawing handler callback (implemented in the Game Activity) to redraw the board. The handler will then call the Drawing Engine to redraw the board. In this sense the drawing engine is a “dumb/slave” system. It doesn’t actually know about the state of the game, it is just given command to move pieces and do other visual activities.



*Figure 1 – Event flow when making a move.*

***Game State Manager***

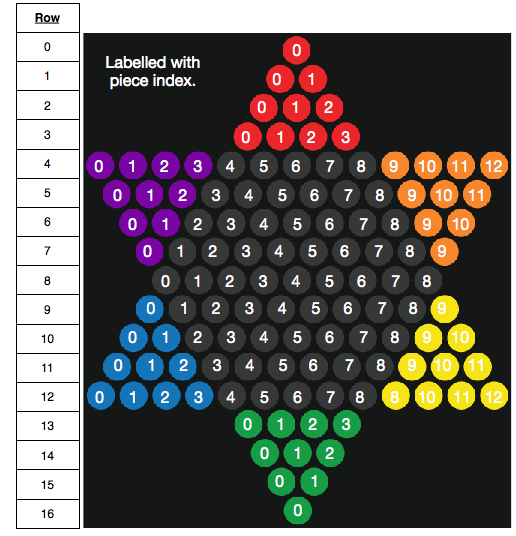
The game state manager manages making sure the state of the game board stays in sync with all other players playing the game. For network-based games this system will involve communicating with the server to send and receive moves to other players. The Hot Seat game play mode which works completely locally works by pretending that it is communicating with the server and handling the move events the same way. Thus those events can be forwarded to the Game Activity without the Game Activity needing much modification to work for both local and network based games.

***Game Board UI Drawing Engine***

The drawing engine receives commands telling it where to move pieces or what to draw from the Game Activity. It then executes those visual commands and redraws the board to reflect the changes.

This engine is also able to detect touch events on any visual element on the game board. Thus the Game Activity can register to be notified when a piece is touched.

The drawing engine expects that all position related information be given to it in row-index form starting at 0. There is an interface called “Position” that should be implemented and follow these constraints.



*Figure 2 – The Row-Index Format*

***Game Activity***

The Game Activity that holds the game board UI is responsible for handling all the intermediary actions to with updating the state of the game board in the Game State Manager and drawing the state onto the screen. This activity knows about the Game State Manger and the Game Board UI Drawing Engine. Using these the activity can receive events from the state manager and then signal to the drawing engine to move or redraw the board based on whatever state change occurred. The game activity can also signal any other drawing event to the drawing engine that it needs to. As well as it will receive touch events on the game board from the drawing engine, which it can then handle appropriately.

**The Game Board**

The game board is responsible for keeping the state of where all the players pieces are in the game. It also validates potential moves so that the player can check if a move is allowed. Every piece is guaranteedto be in a valid location based on our defined game rules. Currently players only have a name associated with them.

**Players**

There are three types of players that implement the “Player” interface.

***Local Player***

This is a player that is in contact with the actual device. A local player in one instance of a game will be a network player for someone else running the application.

***Network Player***

This player instance interacts with the server to get details about another player in the game as they are needed.

***AI Player (Artificial Intelligence)***

The AI player is just an extension of the local player. It can generate custom names as needed.