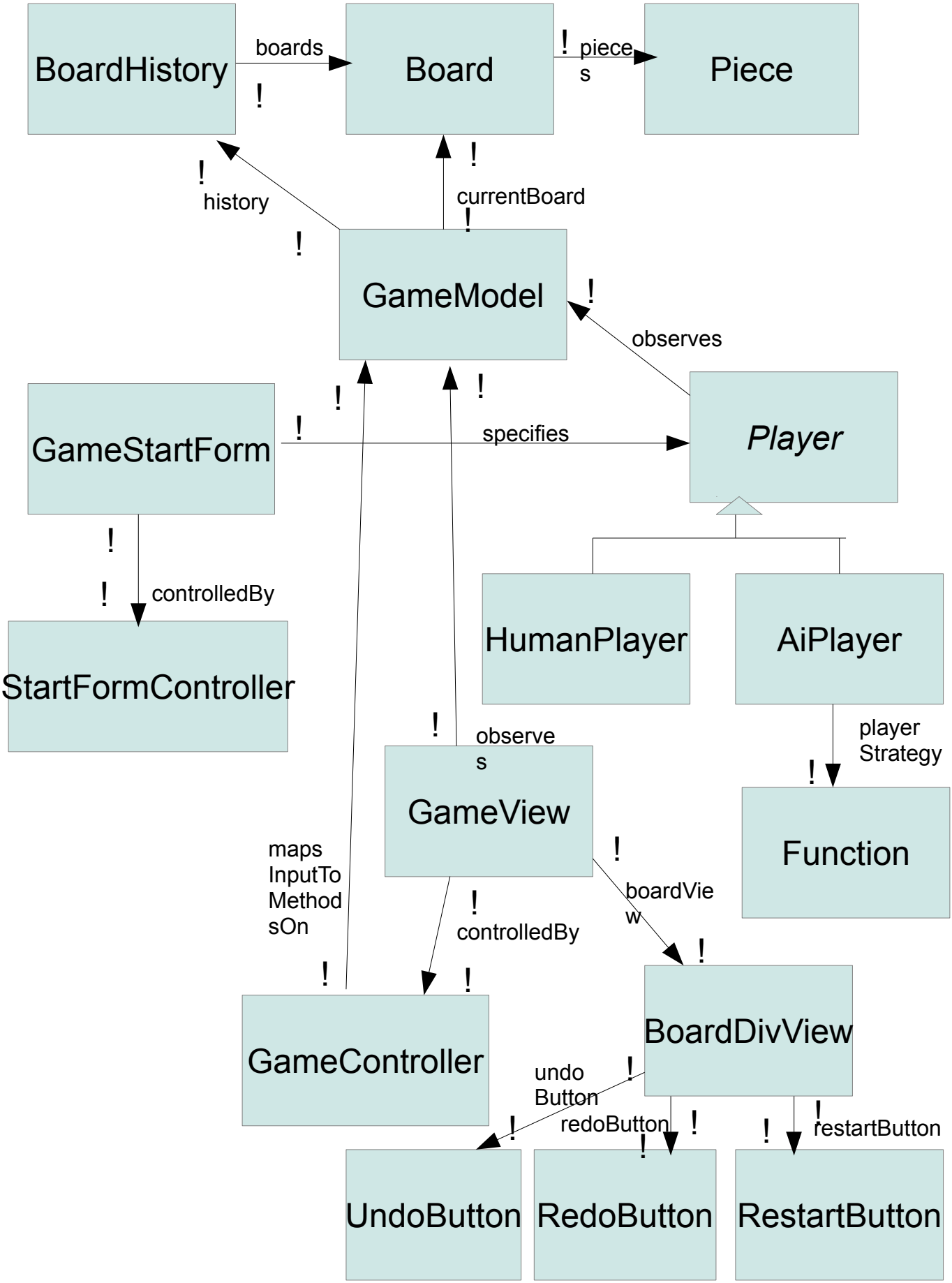
Rahul Rajagopalan
6.170

Othello Object Model



Ambiguities Resolved

The specification of the undo, redo, restart, and abort features were left ambiguous (i.e. how they should be presented and what they should do). My application allows the user to call these by clicking buttons, which are components of BoardDivView. Their behavior is an implementation detail.

Complexities Ignored

The details as to how the objects are wired up in the application are left out of the object model. In the application, this is done through a function in the GameFactory namespace, which creates all the necessary objects to start a game, connects them with each other (primarily through constructor injection), and then starts the game. If it was possible to make the above object graph itself part of an object model, this could be represented by an one-to-many relation between GameFactory and the graph, labeled by “creates”.

This fact also implies that none of the sets listed in the graph are singletons. While the application in its current form only invokes GameFactory once, it would be possible to create multiple games and run them in parallel by calling the function multiple times.

I used two external JavaScript libraries: jQuery for cross-browser DOM interaction, and underscore.js for cross-browser implementations of common functional programming idioms. Their use is considered an implementation detail and is therefore not represented in the object model.

Designations

Boards are immutable objects that represent the state of the game after each turn. When a move is made, a new Board is created and the reference to a Board in GameModel is updated. The old board is added to a GameHistory to support undoing and redoing.

GameModel is a composite object containing a reference to the current Board and a GameHistory. It allows other objects to observe it (using the Observer pattern).

GameStartForm is the form presented to the user at the beginning of the game to specify who is playing. Its input actions are handled by methods on StartFormController.

Players can be either human or AI. They observe the state of the GameModel. Human players ignore changes (instead acting via input to the GUI), and AI players move if it's their turn.

GameViews are composite GUI objects that contain a view of the board in its current state, along with buttons to take various actions. A GameController object handles user input by calling the appropriate methods on the GameModel.