Overview

The Big Dog Poker application allows a user to play poker against computer opponents. Currently the user can play against up to 5 other AI players. The target audience of the application used to be normal poker players, but the target audience is now people who want to learn how to play. As such, we are adding a tutorial mode which will guide players through a round of hold 'em before they start playing for real.

Implementation

The system's menus follow a general MVC model. The model is initialized, and then the views and controllers are initialized on top of the model. Every time the model is updated, model:notify() is called, and that in turn calls observer:update() on all attached observers (aka all controllers and views in our implementation).

The application begins by activating the CharacterSelectionController, which handles picking players. The first player is assumed to be the human player, and every player thereafter is a computer player.

When the user navigates to "Start" and hits Enter/OK on his remote, the game begins. The model passes control to GameControl and gets out of the way. Thereafter, GameControl is for the most part self-sufficient. The entire game engine is designed with an emphasis on event-driven architecture on top of an action pipeline. Every time an event happens, we trigger the GameControl on_event event listener, which runs the action at the front of the pipeline. Depending on the return value of the method call, it will either remove the action from the pipeline, or not. Currently, every stage of the pipeline either starts a timer event, or turns the keyboard listener on, but other events may be added as necessary as well.

The GameControl has GamePresentation and GameState slaves which it directs, while it also calls the lower-level HandControl which has its own presentation and state slaves. These are hooked up with the

Each betting round is composed of player turns, and for each turn, there's a bet setup and a bet execution. The bet setup sets up a listener for the event that a bet is ready (either through the keyboard input interface, or from a timer after a computer has gone). Then the bet execution receives the bet event and triggers a call to HandState, which changes the state of the hand. The human interface utilizes the BettingController (and its corresponding BettingView gives it visibility). The computer move comes from player:get_move(state).

Every time the

Todo

- Fix the bug where chips sometimes remain on the table.

Done

- ☑Implement split pot functionality, so that if the best five-card hand is held simultaneously by two or more players, the pot is split between all the winners. If the pot doesn't split evenly (as will often be the case), then a random subset of players will get an extra dollar (unit of currency). Currently, if there's more than one winner, we just take the first player to be the winner.
- ☑Right now, action for the next betting round is calculated wrong. It takes the dealer (index of dealer in players) and takes it to be the index of dealer in in_players. It's right a lot of the time, and right now the user feedback from the system is so weak that you can't tell there's an issue here, but it'll be obvious when the initial action in certain betting rounds is wrong.
- ☑Sanitize betting input so that it fails hard when the user tries to make an illegal bet.
- ☑Remove a player when he loses all his money.
- ☑Abstracted out initial endowment to a global variable
- ☑Fixed a bug in Player where computer player doesn't actually go all in when he wishes to.
- ☑Clean up the source files of all the extra cruft that doesn't actually run.