9: One Turn

CSCI 6626 / 4526 April 2018

1 Goals

- To implement a menu-based user interface.
- To implement the logic for one player taking one turn.
- To add to your Game class.
- To test the Game class, its interface, and their interaction with the classes PlayerList, Player, Board, Column and Dice.

2 Another Enum.

To your file enums.hpp, add an enumeration for the status of the game: `begun, done`, or `quit`. `Begun` is the initial state, `done` means that some player has won the game, and `quit` is used when the last player resigns.

3 Modify the Game Class

Game data members. Your Game class should have these data members; remove the single players and single columns that were used earlier.

- A Board.
- A CList of Players.
- A Dice* pointing to dynamically allocated FakeDice
- A const int* for the current pair-values of the dice.
- The game state.

Game functions.

- **Constructor and destructor**
  The constructor must initialize all game parts then call getPlayers(). The constructor will be called from `main()`. You will probably not need a print function to debug this class.
- `getPlayers()`
  A private function that will input a list of two to four players from the keyboard, create Player objects, and insert them on the player list. (You might choose to have the number of players passed as a parameter to this function. My own program stops adding players when I enter a / instead of a name.) Get a name and a color for each player. Use the colorNames to print the colors. You must ensure that every player has a different color. (My program does this easily using a mask array. Let me know if you want hints.)
- `void play()`
  A public function called from `main()`. Eventually, it will play one game, start to finish. For this program, simply call `oneTurn()` twice.
- `status oneTurn( Player* p )`
  A private function called from `play()`. It starts, carries out, and finishes a single turn.

  Tell the Board that Player p is starting a new turn, print out the name, color, score and scoreboard for p, then repeat the actions below (in some appropriate order) until the player stops or goes bust or wins or resigns from the game.

  - At the beginning of the turn, present a menu to ask whether the player wants to roll, stop, or resign. If resign is chosen, remove this player from the list of players.
  - If stop is chosen, tell the Board to end the move and call Player::wonColumn() for each column which was captured on this turn.
  - If roll is chosen, call Dice::roll(), which will display the dice to the human, ask the human to choose pairs, and display the pairs.
  - Call Board::move() twice using the two dice-pair values.
  - Go bust if both moves fail.
  - Display the board showing the results of the move.

Testing.

- Test this with three players in the list. Let one of them resign before the first turn. Design a FakeDice input file so that that one player goes bust, the other stops soon enough. One turn each is enough, but be sure the results are clearly displayed.