MelvinLecoy / gitcode Private

Code Issues Pull requests Actions Projects Security Insights Settings

master

gitcode / p3-euker copy / euchre.cpp

Kwan Ting Lau gitcode History

0 contributors

297 lines (263 sloc) | 7.53 KB

```
// Project UID 1d9f47bfc76643019cfbf037641defe1
#include "Pack.h"
#include "Player.h"

#include <algorithm>
#include <cassert>
#include <fstream>
#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Game {
private:
  Card upcard;
  vector<Player *> players = {};
  Player *leader;
  Pack pack;
  int points_to_win = 0;
  int dealer_index = 0;
  bool is_dealer = false;
  string order_up_suit;
  // Team 1: Players 0 and 2.
  // Team 2: Players 1 and 3.
  int ordered_up_team = 0;
  // counter for teams 1 and 2
  int team_1_pts_ctr = 0;
  int team_2_pts_ctr = 0;

public:
  Game(char *argv[]) {
    players.push_back(Player_factory(argv[4], argv[5]));
    players.push_back(Player_factory(argv[6], argv[7]));
    players.push_back(Player_factory(argv[8], argv[9]));
```

```
  players.push_back(Player_factory(argv[10], argv[11]));
  points_to_win = atoi(argv[3]);
}

void init_pack(istream &pack_input, bool shuffle_arg) {
  pack = Pack(pack_input);
  if (shuffle_arg) {
    pack.shuffle();
  }
}

void deal_cards() {
  cout << leader << " deals\n";

  for (int i = dealer_index + 1; i < (dealer_index + 9); i++) {

    if (i == (dealer_index + 1) % 4 || i == (dealer_index + 3) % 4 ||
        i == (dealer_index + 6) % 4 || i == (dealer_index + 8) % 4) {
      for (int j = 0; j < 3; j++) {
        players[i % 4]->add_card(pack.deal_one());
      }
    } else {
      for (int j = 0; j < 2; j++) {
        players[i % 4]->add_card(pack.deal_one());
      }
    }
  }
}

void order_up() {
  upcard = pack.deal_one();
  cout << upcard << " turned up\n";
  Player *p;
  // round 1
  for (int i = (dealer_index + 1); i < (dealer_index + 5); i++) {
    p = players[i % 4];
    if (i % 4 == dealer_index) {
      is_dealer = true;
    }
    if (p->make_trump(upcard, is_dealer, 1, order_up_suit)) {
      if (i % 4 == 0 || i % 4 == 2) {
        ordered_up_team = 1;
      } else if (i % 4 == 1 || i % 4 == 3) {
        ordered_up_team = 2;
      }
      cout << *p << " orders up " << order_up_suit << "\n";
      p->add_and_discard(upcard);
      cout << "\n";
      return;
    } else {
      cout << *p << " passes\n";
    }
  }

```

```
  // round 2
  for (int i = dealer_index + 1; i < (dealer_index + 5); i++) {
    p = players[i % 4];
    if (i % 4 == dealer_index) {
      is_dealer = true;
    }
    if (p->make_trump(upcard, is_dealer, 2, order_up_suit)) {
      if (i % 4 == 0 || i % 4 == 2) {
        ordered_up_team = 1;
      } else if (i % 4 == 1 || i % 4 == 3) {
        ordered_up_team = 2;
      }
      cout << *p << " orders up " << order_up_suit << "\n";
      return;
    } else {
      cout << *p << " passes\n";
    }
  }
}

void play_trick(int trick) {
  Player *current_player;
  Card led_card;
  Card played_card;
  int player_index = 0;
  vector<Card> cards_played = {};
  int trick_taker_index;

  if (trick == 1) {
    leader = players[(dealer_index + 1) % 4];
  }

  led_card = leader->lead_card(order_up_suit);
  cards_played.push_back(led_card);
  cout << led_card << " led by " << leader << "\n";

  for (int i = 1; i < 4; i++) {
    player_index = (dealer_index + i) % 4;
    current_player = players[player_index];
    played_card = current_player->play_card(led_card, order_up_suit);
    cards_played.push_back(played_card);
    cout << played_card << " played by " << current_player << "\n";
    if (Card_less(cards_played[i - 1], cards_played[i], led_card,
                  order_up_suit)) {
      trick_taker_index = player_index;
    }
  }

  leader = players[trick_taker_index];
  cout << leader << " takes the trick\n";
}

void play_hand() {
  int team1_tricks_won = 0;
```

```
    int team2_tricks_won = 0;
    int hand_winner = 0;
    int winner_tricks_won = 0;

    // plays 1 hand (5 tricks)
    for (int i = 1; i < 6; i++) {
      play_trick(i);
      if (leader == players[0] || leader == players[2]) {
        team1_tricks_won++;
      } else if (leader == players[1] || leader == players[3]) {
        team2_tricks_won++;
      }
    }

    if (team1_tricks_won >= 3) {
      hand_winner = 1;
      winner_tricks_won = team1_tricks_won;
    } else if (team2_tricks_won >= 3) {
      hand_winner = 2;
      winner_tricks_won = team2_tricks_won;
    }

    if (hand_winner == ordered_up_team &&
        (winner_tricks_won == 3 || winner_tricks_won == 4)) {
      if (hand_winner == 1) {
        team_1_pts_ctr++;
      } else if (hand_winner == 2) {
        team_2_pts_ctr++;
      }
    } else {
      if (hand_winner == 1) {
        team_1_pts_ctr++;
        team_1_pts_ctr++;
      } else if (hand_winner == 2) {
        team_2_pts_ctr++;
        team_2_pts_ctr++;
      }
    }

    // prints hand winner
    if (hand_winner == 1) {
      cout << players[0] << " and " << players[2] << " win the hand\n";
    } else if (hand_winner == 2) {
      cout << players[1] << " and " << players[3] << " win the hand\n";
    }

    // prints if "euchred" or "march"
    if (hand_winner != ordered_up_team) {
      cout << "euchred!\n";
    } else if (hand_winner == ordered_up_team && winner_tricks_won != 5) {
      cout << "march!\n";
    }

    // prints how many points the teams have
```

```
    cout << players[0] << " and " << players[2] << " have " << team_1_pts_ctr
         << " points\n";

    cout << players[1] << " and " << players[3] << " have " << team_2_pts_ctr
         << " points\n";

    dealer_index = (dealer_index + 1) % 4;
  }

  void play_hands() {
    int i = 0;
    while (true) {
      cout << "Hand " << i;
      deal_cards();
      order_up();
      play_hand();
      if (team_1_pts_ctr >= points_to_win) {
        cout << players[0] << " and " << players[2] << "win!\n";
        break;
      } else if (team_2_pts_ctr >= points_to_win) {
        cout << players[1] << " and " << players[3] << "win!\n";
        break;
      }
      i++;
    }
  }

  ~Game() {
    for (size_t i = 0; i < players.size(); i++) {
      delete players[i];
    }
  }
};

// HELPER FUNCS FOR MAIN BEGIN
//=================================================================

void error_msg() {
  cout << "Usage: euchre.exe pack_input_FILENAME [shuffle_arg|noshuffle_arg] "
       << "POINTS_TO_WIN NAME1 TYPE1 NAME2 TYPE2 NAME3 TYPE3 "
       << "NAME4 TYPE4" << endl;
}

//=================================================================
// HELPER FUNCS FOR MAIN END

int main(int argc, char *argv[]) {
  bool shuffle_arg = false;
  if (argc != 12) {
    error_msg();
    return 1;
  }
  int points_to_win = atoi(argv[3]);
  if (points_to_win < 1 || points_to_win > 100) {
```

```
    error_msg();
    return 1;
  }

  if (string(argv[2]) == "shuffle") {
    shuffle_arg = true;
  } else if (string(argv[2]) == "noshuffle") {
    shuffle_arg = false;
  } else {
    error_msg();
    return 1;
  }

  if (string(argv[5]) != "Simple" && string(argv[5]) == "Human") {
    error_msg();
    return 1;
  }
  if (string(argv[7]) != "Simple" && string(argv[7]) == "Human") {
    error_msg();
    return 1;
  }
  if (string(argv[9]) != "Simple" && string(argv[9]) == "Human") {
    error_msg();
    return 1;
  }
  if (string(argv[11]) != "Simple" && string(argv[11]) == "Human") {
    error_msg();
    return 1;
  }

  ifstream pack_input;
  pack_input.open(argv[1]);
  if (!pack_input.is_open()) {
    cout << "Error opening " << argv[1] << endl;
    return 1;
  }
  // prints all arguments passed in to the main
  for (int i = 0; i < argc; i++) {
    cout << argv[i] << " ";
  }
  cout << "\n";

  Game game(argv);
  game.init_pack(pack_input, shuffle_arg);
  game.play_hands();
}
```

Give feedback