


 [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

...

```
1 // Project UID 1d9f47bfc76643019cfbf037641defe1
2 #include "Pack.h"
3 #include "Player.h"
4
5 #include <algorithm>
6 #include <cassert>
7 #include <fstream>
8 #include <iostream>
9 #include <string>
10 #include <vector>
11
12 using namespace std;
13
14 class Game {
15 private:
16     Card upcard;
17     vector<Player *> players = {};
18     Player *leader;
19     Pack pack;
20     int points_to_win = 0;
21     int dealer_index = 0;
22     bool is_dealer = false;
23     string order_up_suit;
24     // Team 1: Players 0 and 2.
25     // Team 2: Players 1 and 3.
26     int ordered_up_team = 0;
27     // counter for teams 1 and 2
28     int team_1_pts_ctr = 0;
29     int team_2_pts_ctr = 0;
30
31 public:
32     Game(char *argv[]) {
33         players.push_back(Player_factory(argv[4], argv[5]));
34         players.push_back(Player_factory(argv[6], argv[7]));
35         players.push_back(Player_factory(argv[8], argv[9]));
```

```
36     players.push_back(Player_factory(argv[10], argv[11]));
37     points_to_win = atoi(argv[3]);
38 }
39
40 void init_pack(istream &pack_input, bool shuffle_arg) {
41     pack = Pack(pack_input);
42     if (shuffle_arg) {
43         pack.shuffle();
44     }
45 }
46
47 void deal_cards() {
48     cout << leader << " deals\n";
49
50     for (int i = dealer_index + 1; i < (dealer_index + 9); i++) {
51
52         if (i == (dealer_index + 1) % 4 || i == (dealer_index + 3) % 4 ||
53             i == (dealer_index + 6) % 4 || i == (dealer_index + 8) % 4) {
54             for (int j = 0; j < 3; j++) {
55                 players[i % 4]->add_card(pack.deal_one());
56             }
57         } else {
58             for (int j = 0; j < 2; j++) {
59                 players[i % 4]->add_card(pack.deal_one());
60             }
61         }
62     }
63 }
64
65 void order_up() {
66     upcard = pack.deal_one();
67     cout << upcard << " turned up\n";
68     Player *p;
69     // round 1
70     for (int i = (dealer_index + 1); i < (dealer_index + 5); i++) {
71         p = players[i % 4];
72         if (i % 4 == dealer_index) {
73             is_dealer = true;
74         }
75         if (p->make_trump(upcard, is_dealer, 1, order_up_suit)) {
76             if (i % 4 == 0 || i % 4 == 2) {
77                 ordered_up_team = 1;
78             } else if (i % 4 == 1 || i % 4 == 3) {
79                 ordered_up_team = 2;
80             }
81             cout << *p << " orders up " << order_up_suit << "\n";
82             p->add_and_discard(upcard);
83             cout << "\n";
84             return;
85         } else {
86             cout << *p << " passes\n";
87         }
88     }
89 }
```

```
90 // round 2
91 for (int i = dealer_index + 1; i < (dealer_index + 5); i++) {
92     p = players[i % 4];
93     if (i % 4 == dealer_index) {
94         is_dealer = true;
95     }
96     if (p->make_trump(upcard, is_dealer, 2, order_up_suit)) {
97         if (i % 4 == 0 || i % 4 == 2) {
98             ordered_up_team = 1;
99         } else if (i % 4 == 1 || i % 4 == 3) {
100             ordered_up_team = 2;
101         }
102         cout << *p << " orders up " << order_up_suit << "\n";
103         return;
104     } else {
105         cout << *p << " passes\n";
106     }
107 }
108 }
109
110 void play_trick(int trick) {
111     Player *current_player;
112     Card led_card;
113     Card played_card;
114     int player_index = 0;
115     vector<Card> cards_played = {};
116     int trick_taker_index;
117
118     if (trick == 1) {
119         leader = players[(dealer_index + 1) % 4];
120     }
121
122     led_card = leader->lead_card(order_up_suit);
123     cards_played.push_back(led_card);
124     cout << led_card << " led by " << leader << "\n";
125
126     for (int i = 1; i < 4; i++) {
127         player_index = (dealer_index + i) % 4;
128         current_player = players[player_index];
129         played_card = current_player->play_card(led_card, order_up_suit);
130         cards_played.push_back(played_card);
131         cout << played_card << " played by " << current_player << "\n";
132         if (Card_less(cards_played[i - 1], cards_played[i], led_card,
133             order_up_suit)) {
134             trick_taker_index = player_index;
135         }
136     }
137
138     leader = players[trick_taker_index];
139     cout << leader << " takes the trick\n";
140 }
141
142 void play_hand() {
143     int team1_tricks_won = 0;
```

```
144     int team2_tricks_won = 0;
145     int hand_winner = 0;
146     int winner_tricks_won = 0;
147
148     // plays 1 hand (5 tricks)
149     for (int i = 1; i < 6; i++) {
150         play_trick(i);
151         if (leader == players[0] || leader == players[2]) {
152             team1_tricks_won++;
153         } else if (leader == players[1] || leader == players[3]) {
154             team2_tricks_won++;
155         }
156     }
157
158     if (team1_tricks_won >= 3) {
159         hand_winner = 1;
160         winner_tricks_won = team1_tricks_won;
161     } else if (team2_tricks_won >= 3) {
162         hand_winner = 2;
163         winner_tricks_won = team2_tricks_won;
164     }
165
166     if (hand_winner == ordered_up_team &&
167         (winner_tricks_won == 3 || winner_tricks_won == 4)) {
168         if (hand_winner == 1) {
169             team_1_pts_ctr++;
170         } else if (hand_winner == 2) {
171             team_2_pts_ctr++;
172         }
173     } else {
174         if (hand_winner == 1) {
175             team_1_pts_ctr++;
176             team_1_pts_ctr++;
177         } else if (hand_winner == 2) {
178             team_2_pts_ctr++;
179             team_2_pts_ctr++;
180         }
181     }
182
183     // prints hand winner
184     if (hand_winner == 1) {
185         cout << players[0] << " and " << players[2] << " win the hand\n";
186     } else if (hand_winner == 2) {
187         cout << players[1] << " and " << players[3] << " win the hand\n";
188     }
189
190     // prints if "euchred" or "march"
191     if (hand_winner != ordered_up_team) {
192         cout << "euchred!\n";
193     } else if (hand_winner == ordered_up_team && winner_tricks_won != 5) {
194         cout << "march!\n";
195     }
196
197     // prints how many points the teams have
```

```
198     cout << players[0] << " and " << players[2] << " have " << team_1_pts_ctr
199         << " points\n";
200
201     cout << players[1] << " and " << players[3] << " have " << team_2_pts_ctr
202         << " points\n";
203
204     dealer_index = (dealer_index + 1) % 4;
205 }
206
207 void play_hands() {
208     int i = 0;
209     while (true) {
210         cout << "Hand " << i;
211         deal_cards();
212         order_up();
213         play_hand();
214         if (team_1_pts_ctr >= points_to_win) {
215             cout << players[0] << " and " << players[2] << "win!\n";
216             break;
217         } else if (team_2_pts_ctr >= points_to_win) {
218             cout << players[1] << " and " << players[3] << "win!\n";
219             break;
220         }
221         i++;
222     }
223 }
224
225 ~Game() {
226     for (size_t i = 0; i < players.size(); i++) {
227         delete players[i];
228     }
229 }
230 };
231
232 // HELPER FUNCS FOR MAIN BEGIN
233 //=====
234
235 void error_msg() {
236     cout << "Usage: euchre.exe pack_input_FILENAME [shuffle_arg|noshuffle_arg] "
237         << "POINTS_TO_WIN NAME1 TYPE1 NAME2 TYPE2 NAME3 TYPE3 "
238         << "NAME4 TYPE4" << endl;
239 }
240
241 //=====
242 // HELPER FUNCS FOR MAIN END
243
244 int main(int argc, char *argv[]) {
245     bool shuffle_arg = false;
246     if (argc != 12) {
247         error_msg();
248         return 1;
249     }
250     int points_to_win = atoi(argv[3]);
251     if (points_to_win < 1 || points_to_win > 100) {
```

```
252     error_msg();
253     return 1;
254 }
255
256 if (string(argv[2]) == "shuffle") {
257     shuffle_arg = true;
258 } else if (string(argv[2]) == "noshuffle") {
259     shuffle_arg = false;
260 } else {
261     error_msg();
262     return 1;
263 }
264
265 if (string(argv[5]) != "Simple" && string(argv[5]) == "Human") {
266     error_msg();
267     return 1;
268 }
269 if (string(argv[7]) != "Simple" && string(argv[7]) == "Human") {
270     error_msg();
271     return 1;
272 }
273 if (string(argv[9]) != "Simple" && string(argv[9]) == "Human") {
274     error_msg();
275     return 1;
276 }
277 if (string(argv[11]) != "Simple" && string(argv[11]) == "Human") {
278     error_msg();
279     return 1;
280 }
281
282 ifstream pack_input;
283 pack_input.open(argv[1]);
284 if (!pack_input.is_open()) {
285     cout << "Error opening " << argv[1] << endl;
286     return 1;
287 }
288 // prints all arguments passed in to the main
289 for (int i = 0; i < argc; i++) {
290     cout << argv[i] << " ";
291 }
292 cout << "\n";
293
294 Game game(argv);
295 game.init_pack(pack_input, shuffle_arg);
296 game.play_hands();
297 }
```

[Give feedback](#)