```jsx
/**
 * @file Game.jsx
 * @description React extension javascript that exports a Game react component.
 * @author Manyi Cheng
 */

import React, { Component } from 'react';
import Player from './Player.jsx';
import Deck from './Deck.jsx';
import GameplayField from './GameplayField.jsx';
import peachIcon from '../res/peach.png';
import luigiIcon from '../res/luigi.png';
import booIcon from '../res/boo.png';
import Timer from './Timer.js';
import * as Rules from '../Rules.js';
import * as PlayerBot from '../PlayerBot.js';
import startButton from '../res/startbutton.png';


/**
 * @class A class that extends react Component, represents a big two Game.
 * @description This class represents Game component in a big two game.
 * @param {*} props Props from parent component.
 */
class Game extends Component {
        constructor(props) {
                super(props);
                this.state = {
                        rules: true,
                        playerScore: 0,
                        playerCards: [],
                        leftCards: [],
                        topCards: [],
                        rightCards: [],
                        playerField: [],
                        leftField: [],
                        topField: [],
                        rightField: [],
                        startingTurn: true,
                        turn: null,
                        minutes: 10,
                        seconds: 0,
                        cardsPlayed: [],
                        freeMove: false,
                        lastMove: [],
                        lastMovePlayer: null,
                        gameOver: false,
                };
                this.startGame = this.startGame.bind(this);
                this.resetGame = this.resetGame.bind(this);
                this.handlePlayerDeal = this.handlePlayerDeal.bind(this);
                this.handlePlayerPass = this.handlePlayerPass.bind(this);
                this.BotPlayCards = this.BotPlayCards.bind(this);
                this.updateNextTurn = this.updateNextTurn.bind(this);
                this.updateField = this.updateField.bind(this);
                this.updateNextTurnCards = this.updateNextTurnCards.bind(this);
                this.getCardsforTurn = this.getCardsforTurn.bind(this);
                this.typeSort = this.typeSort.bind(this);
```

```
59                  this.handleTimer = this.handleTimer.bind(this);
60                  this.suitSort = this.suitSort.bind(this);
61                  this.isGameOver = this.isGameOver.bind(this);
62                  this.displayPass = this.displayPass.bind(this);
63          }
64
65          /**
66           * @description Execute the code synchronously when the component gets loac
    mounted in the DOM. This method is called during the mounting phase of the React Li
67           * @deprecated Will be decrecated be React in the future.
68           */
69          UNSAFE_componentWillMount() {
70                  this.resetGame();
71          }
72
73          /**
74           * @description Starts the game upon user closing the rules.
75           */
76          startGame() {
77                  this.setState({
78                          rules: false,
79                  });
80                  if (this.state.turn !== 'player') {
81                          this.BotPlayCards();
82                  }
83          }
84
85          /**
86           * @description Resets game states upon user clicking play again button.
87           */
88          async resetGame() {
89                  let deck = Rules.newDeck();
90
91                  let playerCards = await Rules.setUserCards(deck);
92                  let leftCards = await Rules.setUserCards(deck);
93                  let topCards = await Rules.setUserCards(deck);
94                  let rightCards = await Rules.setUserCards(deck);
95
96                  let turn = Rules.setFirstTurn(playerCards, leftCards, topCards, rig
97
98                  this.setState({
99                          rules: true,
100                         playerScore: 0,
101                         playerField: [],
102                         leftField: [],
103                         topField: [],
104                         rightField: [],
105                         playerCards: playerCards,
106                         leftCards: leftCards,
107                         topCards: topCards,
108                         rightCards: rightCards,
109                         initialMinutes: 10,
110                         initialSeconds: 0,
111                         turn: turn,
112                         startingTurn: true,
113                         cardsPlayed: [],
114                         lastMove: [],
115                         lastMovePlayer: null,
116                         gameOver: false,
```

```
117                         playerFieldText: '',
118                 });
119         }
120
121         /**
122          * Handles game over condition when the timer reaches 0.
123          */
124         handleTimer() {
125                 this.setState({
126                         gameOver: true,
127                 });
128         }
129
130         /**
131          * @description player action on clicking deal button with selected cards.
132          * @param {*} cards Selected cards to be dealt.
133          * @returns true if valid play, false if invalid play.
134          */
135         handlePlayerDeal(cards) {
136                 this.setState({ playerFieldText: '' });
137                 if (this.state.startingTurn) {
138                         let validPlay = Rules.isValidStartingPlay(cards);
139
140                         if (validPlay) {
141                                 this.updateNextTurnCards(cards);
142                                 this.setState({ startingTurn: false });
143                                 return true;
144                         } else {
145                                 this.setState({
146                                         playerFieldText: 'Your play must be valid a
    3 of diamonds for starting turn',
147                                 });
148                         }
149                 } else {
150                         let valid = Rules.isValidPlay(cards);
151                         let isFreeMove = this.state.lastMovePlayer === 'player';
152                         let stronger = Rules.isStrongerPlay(this.state.lastMove, ca
153
154                         if (valid && (isFreeMove || stronger)) {
155                                 this.updateNextTurnCards(cards);
156                                 return true;
157                         } else {
158                                 if (!valid) {
159                                         this.setState({
160                                                 playerFieldText: 'Your play must be
161                                         });
162                                 } else if (!stronger && cards.length ===
    this.state.lastMove.length) {
163                                         this.setState({ playerFieldText: 'Your play
    stronger than the previous play' });
164                                 } else if (cards.length !== this.state.lastMove) {
165                                         this.setState({
166                                                 playerFieldText: 'Your play must co
    number of cards as the previous play',
167                                         });
168                                 }
169                         }
170                 }
171         }
```

```
172
173         /**
174          * @description Controls the logic when its bot's turn to play cards.
175          */
176         BotPlayCards() {
177                 let currentCards = this.getCardsforTurn();
178                 let bestMove;
179
180                 if (this.state.startingTurn) {
181                         bestMove = PlayerBot.BotStartingTurn(currentCards);
182                         this.setState({ startingTurn: false });
183                 } else {
184                         if (this.state.lastMovePlayer === this.state.turn) {
185                                 bestMove = PlayerBot.BotFreeTurn(currentCards);
186                         } else {
187                                 bestMove = PlayerBot.BotPlayCards(currentCards,
    this.state.lastMove);
188                         }
189                 }
190
191                 this.updateNextTurnCards(bestMove);
192         }
193
194         /**
195          * @description gets the current players' cards of the turn.
196          * @returns current player cards
197          */
198         getCardsforTurn() {
199                 if (this.state.turn === 'left') return this.state.leftCards;
200                 if (this.state.turn === 'top') return this.state.topCards;
201                 if (this.state.turn === 'right') return this.state.rightCards;
202                 if (this.state.turn === 'player') return this.state.playerCards;
203         }
204
205         /**
206          * @description Updates state cards for next turn based on the cards dealt
    current player.
207          * @param {*} cards Cards dealt by the current player.
208          */
209         updateNextTurnCards(cards) {
210                 if (cards) {
211                         let cardsPlayed = this.state.cardsPlayed;
212                         let currentPlayerCards = this.getCardsforTurn();
213
214                         cards.forEach((card) => {
215                                 currentPlayerCards.splice(currentPlayerCards.indexC
    1);
216                         });
217
218                         if (this.state.lastMove) {
219                                 this.state.lastMove.forEach((card) => {
220                                         cardsPlayed.push(card);
221                                 });
222                         }
223
224                         if (this.state.turn === 'left') this.setState({ leftCards:
    currentPlayerCards });
225                         if (this.state.turn === 'top') this.setState({ topCards:
    currentPlayerCards });
```

```
226                         if (this.state.turn === 'right') this.setState({ rightCards
currentPlayerCards });
227                         if (this.state.turn === 'player') this.setState({ playerCar
currentPlayerCards });
228
229                         this.updateField(cards);
230
231                         this.setState(
232                                 {
233                                         cardsPlayed: cardsPlayed,
234                                         lastMove: cards,
235                                         freeMove: false,
236                                         lastMovePlayer: this.state.turn,
237                                 },
238                                 () => {
239                                         this.updateNextTurn();
240                                 }
241                         );
242                 } else {
243                         if (this.state.turn === 'left')
244                                 this.setState({ leftField: [] }, () => {
245                                         this.displayPass();
246                                 });
247                         if (this.state.turn === 'top')
248                                 this.setState({ topField: [] }, () => {
249                                         this.displayPass();
250                                 });
251                         if (this.state.turn === 'right')
252                                 this.setState({ rightField: [] }, () => {
253                                         this.displayPass();
254                                 });
255                         if (this.state.turn === 'player')
256                                 this.setState({ playerField: [] }, () => {
257                                         this.displayPass();
258                                 });
259
260                         this.updateNextTurn();
261                 }
262         }
263
264         /**
265          * @description Updates the GamplayField when players deal cards.
266          * @param {*} cards Field cards
267          */
268         updateField(cards) {
269                 if (this.state.turn === 'left')
270                         this.setState({ leftField: [] }, () => {
271                                 this.setState({ leftField: cards });
272                         });
273                 if (this.state.turn === 'top')
274                         this.setState({ topField: [] }, () => {
275                                 this.setState({ topField: cards });
276                         });
277                 if (this.state.turn === 'right')
278                         this.setState({ rightField: [] }, () => {
279                                 this.setState({ rightField: cards });
280                         });
281                 if (this.state.turn === 'player')
282                         this.setState({ playerField: [] }, () => {
```

```
283                                 this.setState({ playerField: cards });
284                         });
285         }
286
287         /**
288          * @description Set states turn, and field text for next turn, then on call
    triggers next turn's play.
289          * @returns Nothing
290          */
291         updateNextTurn() {
292                 if (this.isGameOver()) return;
293                 setTimeout(() => {
294                         if (this.state.turn === 'player') {
295                                 this.setState({ turn: 'right', playerFieldText: ''
296                                         this.BotPlayCards();
297                                 });
298                         } else if (this.state.turn === 'right') {
299                                 this.setState({ turn: 'top' }, () => {
300                                         this.BotPlayCards();
301                                 });
302                         } else if (this.state.turn === 'top') {
303                                 this.setState({ turn: 'left' }, () => {
304                                         this.BotPlayCards();
305                                 });
306                         } else this.setState({ turn: 'player' });
307                 }, 1200);
308         }
309
310         /**
311          * @description Handles player passing for starting turn, last move, free m
    normal situations.
312          */
313         handlePlayerPass() {
314                 if (this.state.startingTurn) {
315                         this.setState({
316                                 freeMove: true,
317                                 playerFieldText: 'You cannot pass the first turn',
318                         });
319                 } else if (this.state.lastMovePlayer === 'player') {
320                         this.setState({
321                                 freeMove: true,
322                                 playerFieldText: 'You cannot pass the free move',
323                         });
324                 } else {
325                         this.setState({ playerField: [], playerFieldText: '' });
326                         this.displayPass();
327                         this.updateNextTurn();
328                 }
329         }
330
331         /**
332          * @description Sorts player's cards in type order upon player clicking typ
333          */
334         typeSort() {
335                 let cards = this.state.playerCards;
336                 Rules.sortCardsValue(cards);
337
338                 this.setState({ playerCards: cards });
339         }
```

```
340         /**
341          * @description Sorts player's cards in suit order upon player clicking sui
342          */
343         suitSort() {
344                 let cards = this.state.playerCards;
345                 Rules.sortCardsSuit(cards);
346 
347                 this.setState({ playerCards: cards });
348         }
349 
350         /**
351          * @description Checks whether the game is over and sets the game states ga
    playerScore 1s after validation.
352          */
353         isGameOver() {
354                 let currentPlayerCards = this.getCardsforTurn();
355                 if (currentPlayerCards.length === 0) {
356                         let score = this.computePlayerScore();
357                         setTimeout(() => {
358                                 this.setState({
359                                         gameOver: true,
360                                         playerScore: score,
361                                 });
362                                 return true;
363                         }, 1000);
364                 }
365         }
366 
367         /**
368          * @description Computes player score of the game.
369          * @returns {int} Computed score.
370          */
371         computePlayerScore() {
372                 let len = this.state.playerCards.length;
373                 return Math.ceil((13 - len) * (100 / 13));
374         }
375 
376         /**
377          * @description Displays text when players choose to pass the current turn.
378          */
379         displayPass() {
380                 let field = this.state.turn;
381                 let node = document.createElement('div');
382                 node.append(document.createTextNode('Pass'));
383                 node.setAttribute('class', 'gameplayfield-text');
384                 document.getElementById(field).append(node);
385                 setTimeout(() => {
386                         document.getElementById(field).removeChild(node);
387                 }, 1000);
388         }
389 
390         render() {
391                 if (this.state.rules) {
392                         return (
393                                 <div>
394                                         <div className="game-container">
395                                                 <div className="window-container">
396                                                         <div className="window">
397                                                                 <div className="rul
```

```
                                                                        <h4 classNa
heading">
                                                                            <sp
className="rules-heading-span">Rules</span>
                                                                        </h4>
                                                                </div>
                                                                <div className="rul
details">
                                                                        <ul classNa
details">
                                                                            <li
A > K > Q > J > 10 > 9 > 8 > 7 > 6 > 5 > 4 > 3 </li>
                                                                            <li
Spades > hearts > clubs > diamonds</li>
                                                                            <li
combinations: single, pairs, triples, five-cards</li>
                                                                            <li

combination can only be beaten by a better combination with the same

of cards.
                                                                            </l
                                                                            <li
                                                                            <li
consists of five cards of consecutive rank with mixed suits.</li>
                                                                            <li
consists of any five cards of the same suit.</li>
                                                                            <li

House consists of three cards of one rank and two of another rank
                                                                            </l
                                                                            <li
made up of all four cards of one rank, plus any fifth card</li>
                                                                            <li
Flush consists of five consecutive cards of the same suit.</li>
                                                                        </ul>
                                                                </div>
                                                                <div className="rul
                                                                        <img

,
className="start-button"

,
src={startButton}
                                                                            onC
,
{this.startGame}
                                                                            alt
,
button"
                                                                        />
                                                                </div>
                                                                <div>3XA3 G06</div>
                                                        </div>
                                                </div>
                                        </div>
                                </div>
                        );
                } else {
                        return (
                                <div>
                                        <div className="game-container">
                                                {this.state.gameOver && <div
className="window-container">
                                                        <div className="window">
```

```
                                <div className="gam
container">
                                    <div>Game C
                                    <div>Score
{this.state.playerScore}</div>
                                    <button
                                        id=
,
button"
                                        dis
,
{false}
,
className="playagain-button"
                                        onC
,
{this.resetGame}
,
                                    >
                                        Pla
                                    </button>
                                </div>
                            </div>
                        </div>}
                        <div className="game-opponent">
                            <img src={booIcon} alt="cha
className="top-icon" />
                            <img src={luigiIcon} alt="c
className="opponent-icon" />
                            <div className="game-left">
                                <Deck
                                    class="oppo
,
container-left"
                                    cardClass="
,
side"
                                    cards=
,
{this.state.leftCards}
,
                                ></Deck>
                            </div>
                            <div className="game-middle
                                <Deck
                                    class="oppo
,
container-top"
                                    cardClass="
,
top"
                                    cards=
,
{this.state.topCards}
,
                                ></Deck>
                                <GameplayField
                                    player=
,
{this.state.playerField}
                                    right=
,
{this.state.rightField}
,
left={this.state.leftField}
,
top={this.state.topField}
                                    playerField
,
{this.state.playerFieldText}
,
                                ></GameplayField>
                            </div>
                            <div className="game-right"
                                <Timer
                                    initialMinu
,
{this.state.minutes}
```

```
482 ,                                                                   initialSecc
    {this.state.seconds}
483 ,                                                                   onTimer=
    {this.handleTimer}
484 ,                                                                 />
485                                                                   <Deck
486 ,                                                                   class="oppc
    container-right"
487 ,                                                                   cardClass="
    side"
488 ,                                                                   cards=
    {this.state.rightCards}
489 ,                                                                 ></Deck>
490                                                               </div>
491                                                               <img src={peachIcon} alt="c
    className="opponent-icon" />
492                                                         </div>
493                                                         <Player
494 ,                                                               cards={this.state.playerCar
495 ,                                                               playerTurn={this.state.turn
    'player'}
496 ,                                                               freeMove={this.state.freeMc
497 ,                                                               playCards={this.handlePlaye
498 ,                                                               passTurn={this.handlePlayer
499 ,                                                               turn={this.state.turn}
500 ,                                                               typeSort={this.typeSort}
501 ,                                                               suitSort={this.suitSort}
502 ,                                                               gameOver={this.state.gameOv
503 ,                                                               playerScore={this.state.pla
504 ,                                                         ></Player>
505                                                   </div>
506                                             </div>
507                                 );
508                             }
509                     }
510 }
511
512 export default Game;
```